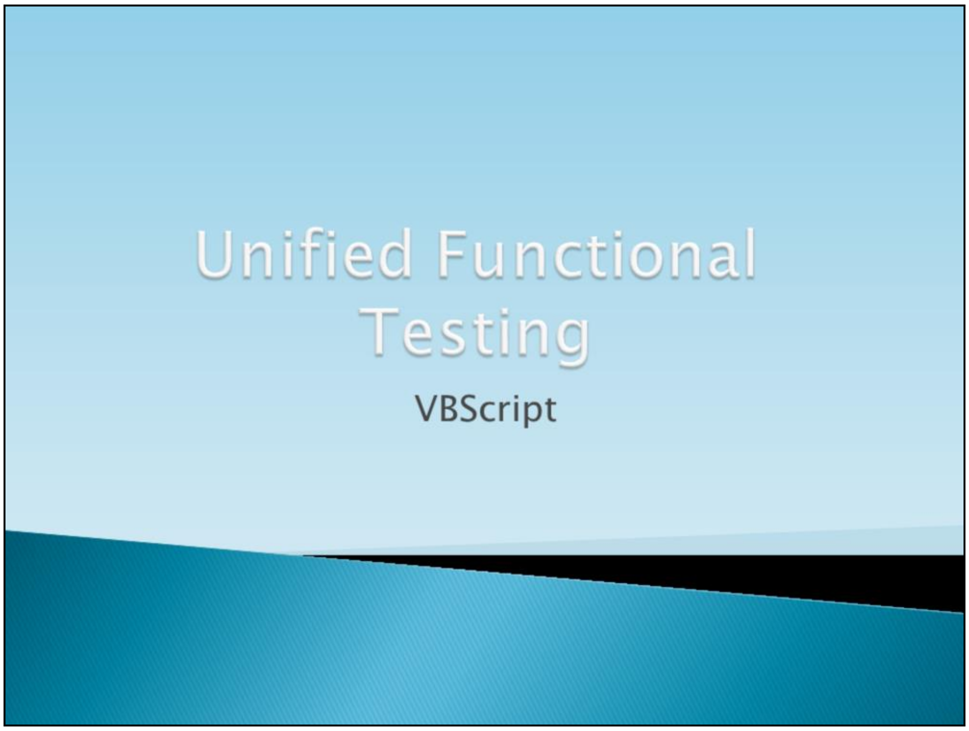
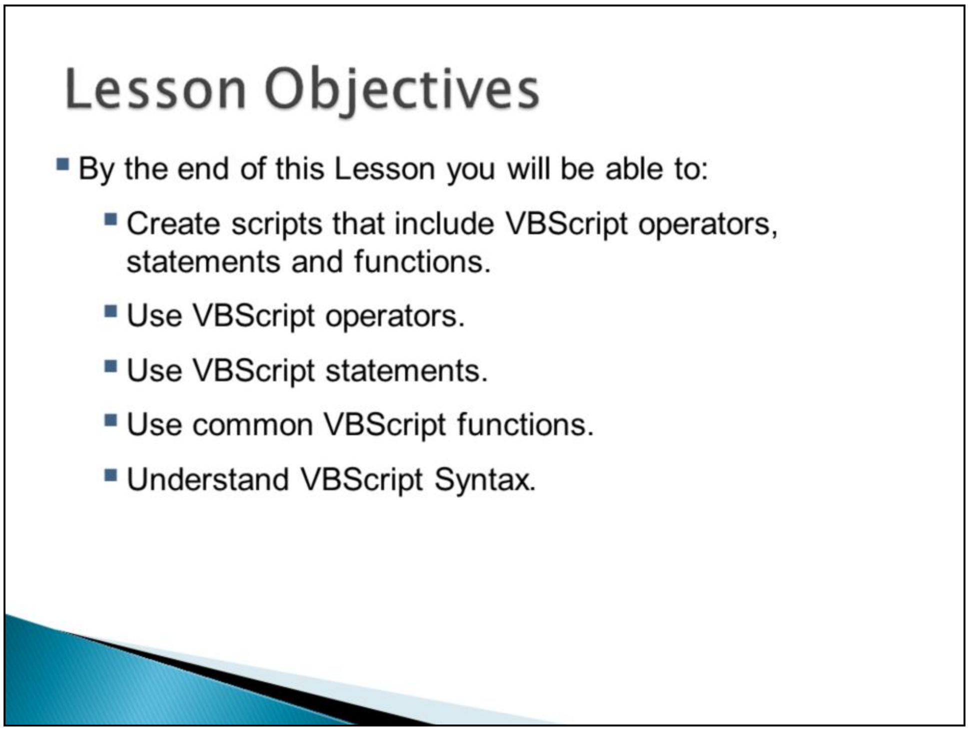


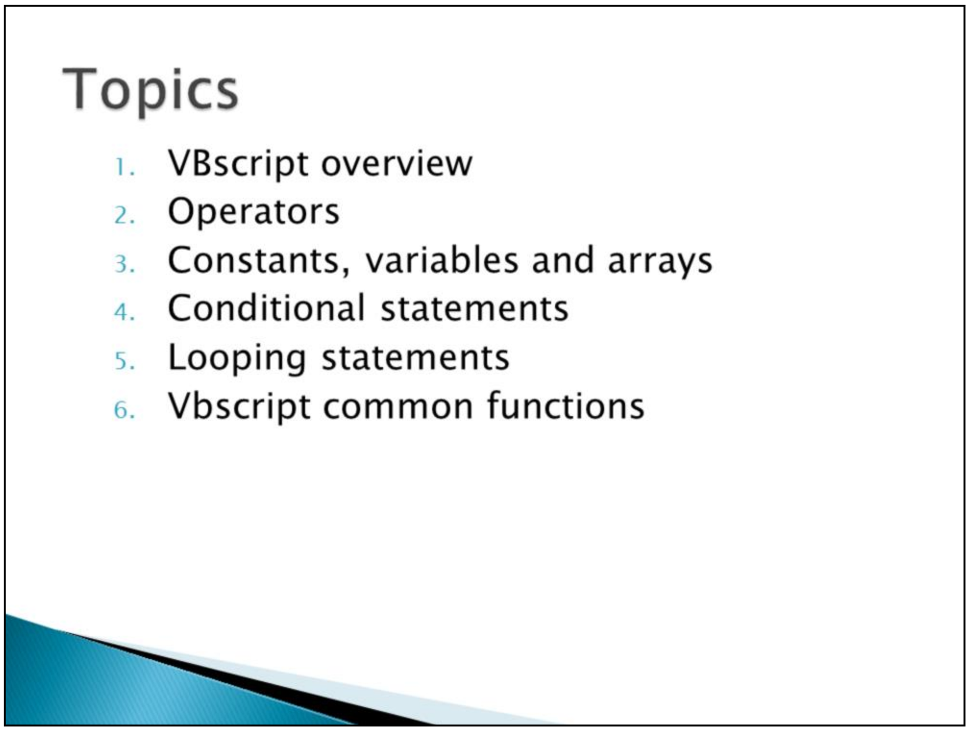
Unified Functional Testing

VBScript



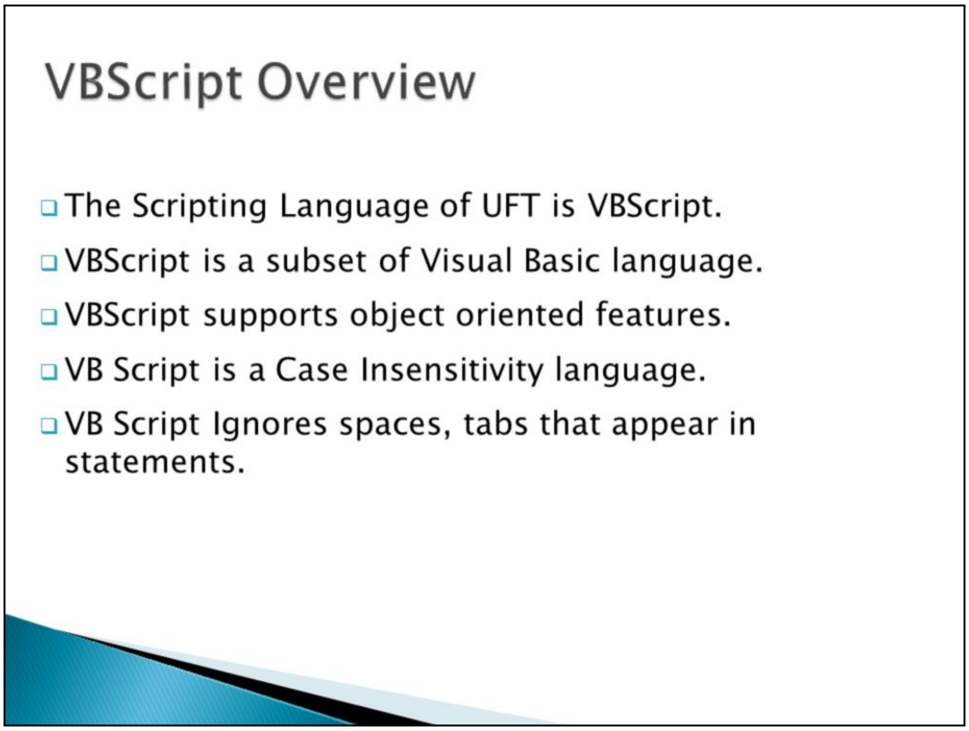
Lesson Objectives

- By the end of this Lesson you will be able to:
  - Create scripts that include VBScript operators, statements and functions.
  - Use VBScript operators.
  - Use VBScript statements.
  - Use common VBScript functions.
  - Understand VBScript Syntax.



Topics

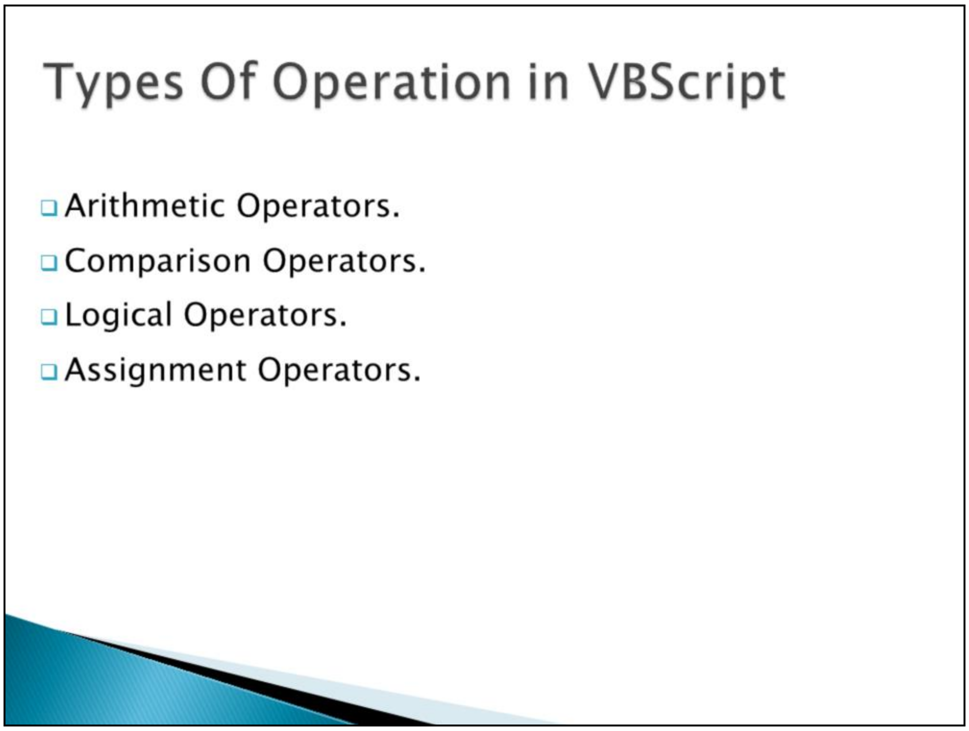
1. VBscript overview
2. Operators
3. Constants, variables and arrays
4. Conditional statements
5. Looping statements
6. Vbscript common functions



VBScript Overview

- The Scripting Language of UFT is VBScript.
- VBScript is a subset of Visual Basic language.
- VBScript supports object oriented features.
- VB Script is a Case Insensitivity language.
- VB Script Ignores spaces, tabs that appear in statements.

In our script we using test objects and methods , in addition we can use also use standard VBScript statements.



Types Of Operation in VBScript

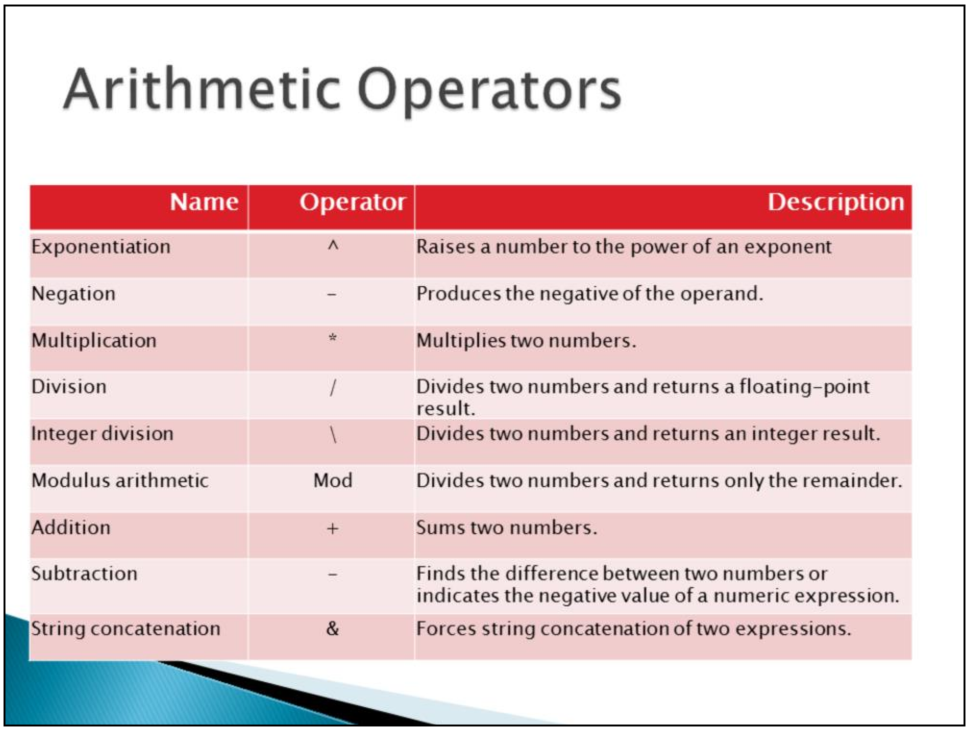
- Arithmetic Operators.
- Comparison Operators.
- Logical Operators.
- Assignment Operators.

Arithmetic Operators - Operators used to perform mathematical calculations and Operators used to combine strings.

Comparison Operators-Operators used to perform comparisons.

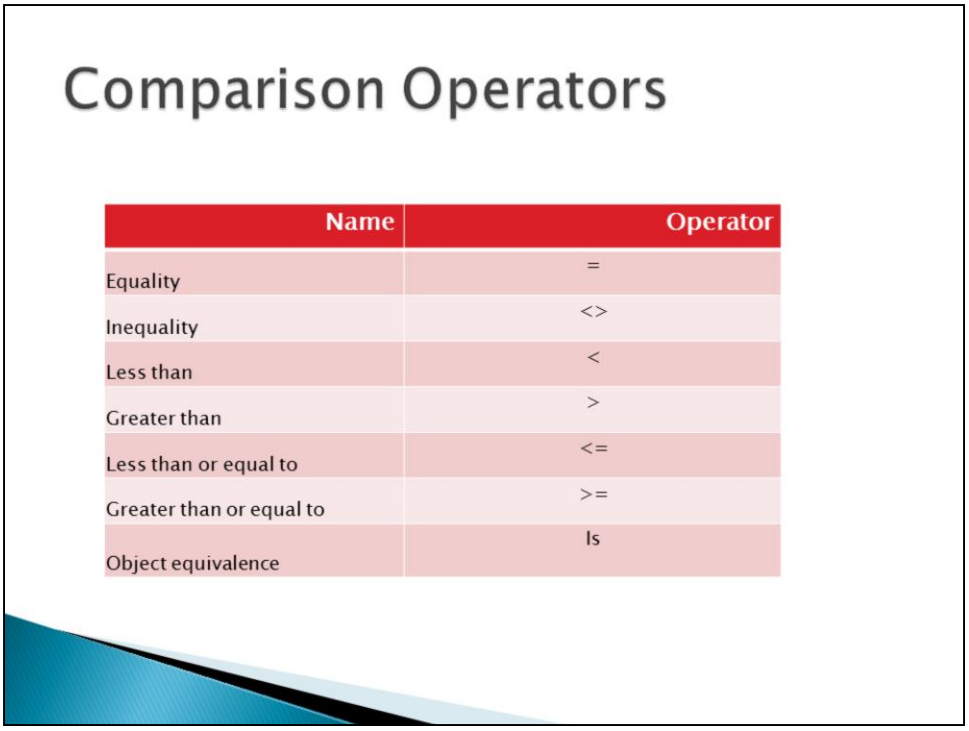
Logical Operators-Operators used to perform logical operations.

Assignment Operators-Operator used to assign a value to a property or variable.



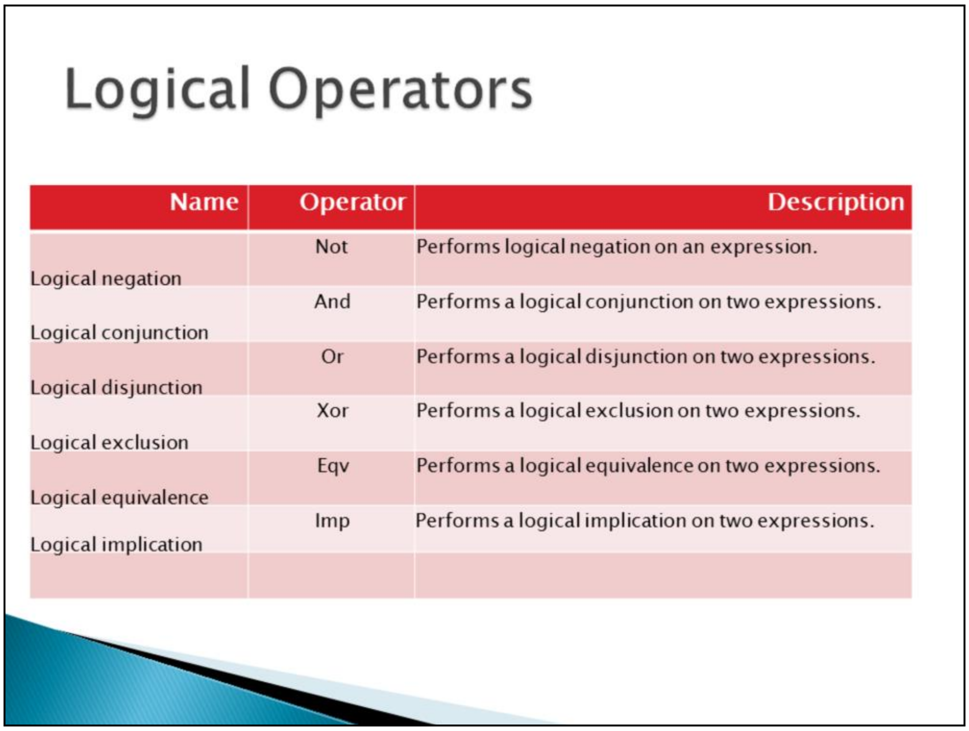
Arithmetic Operators

Name	Operator	Description
Exponentiation	^	Raises a number to the power of an exponent
Negation	–	Produces the negative of the operand.
Multiplication	*	Multiplies two numbers.
Division	/	Divides two numbers and returns a floating-point result.
Integer division	\	Divides two numbers and returns an integer result.
Modulus arithmetic	Mod	Divides two numbers and returns only the remainder.
Addition	+	Sums two numbers.
Subtraction	–	Finds the difference between two numbers or indicates the negative value of a numeric expression.
String concatenation	&	Forces string concatenation of two expressions.



Comparison Operators

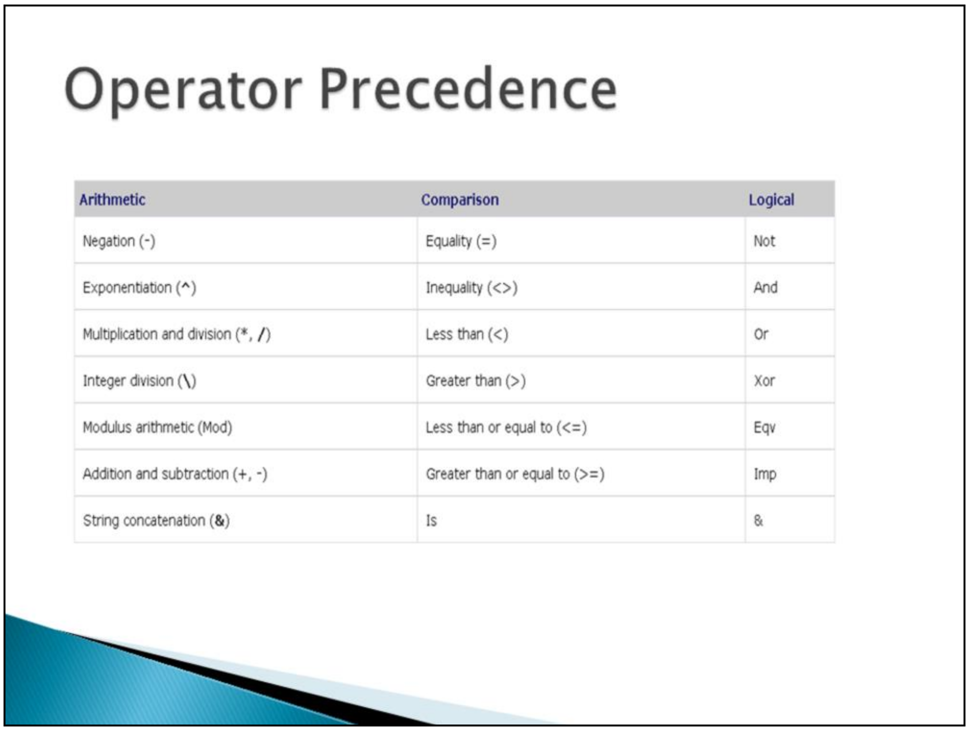
Name	Operator
Equality	=
Inequality	<>
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Object equivalence	Is



Logical Operators

Name	Operator	Description
Logical negation	Not	Performs logical negation on an expression.
Logical conjunction	And	Performs a logical conjunction on two expressions.
Logical disjunction	Or	Performs a logical disjunction on two expressions.
Logical exclusion	Xor	Performs a logical exclusion on two expressions.
Logical equivalence	Eqv	Performs a logical equivalence on two expressions.
Logical implication	Imp	Performs a logical implication on two expressions.

Xor - Dfference



Operator Precedence

Arithmetic	Comparison	Logical
Negation (-)	Equality (=)	Not
Exponentiation (^)	Inequality (<>)	And
Multiplication and division (*, /)	Less than (<)	Or
Integer division (\)	Greater than (>)	Xor
Modulus arithmetic (Mod)	Less than or equal to (<=)	Eqv
Addition and subtraction (+, -)	Greater than or equal to (>=)	Imp
String concatenation (&)	Is	&

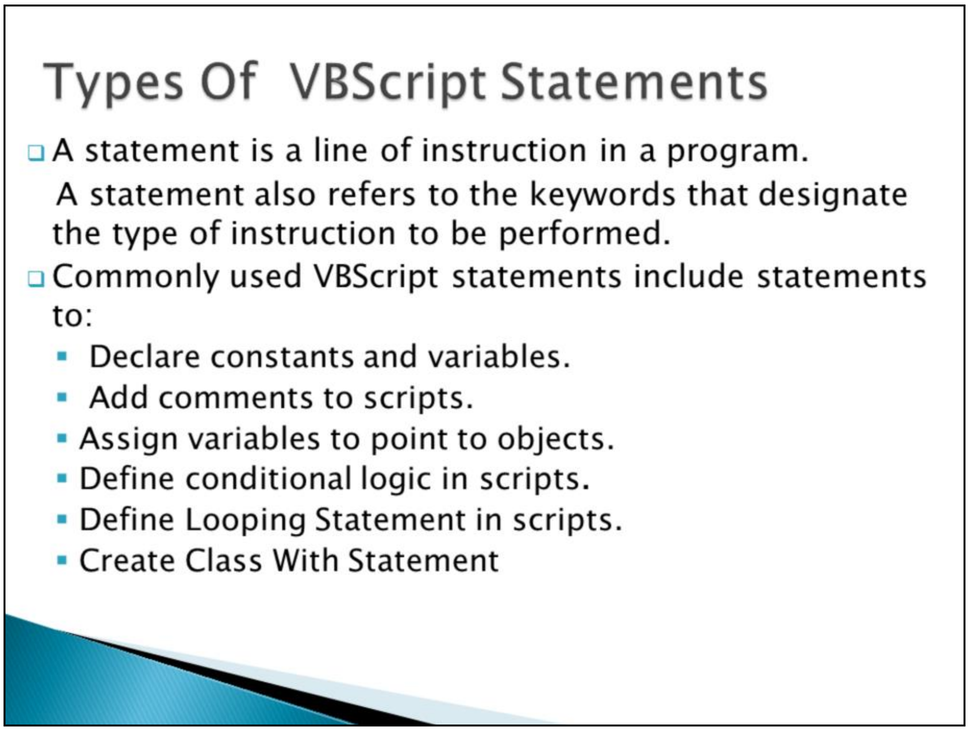
When expressions contain operators from more than one category, **arithmetic operators** are evaluated first, **comparison operators** are evaluated next, and **logical operators** are evaluated last.

Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

Arithmetic and logical operators are evaluated as appear in above order of precedence table.

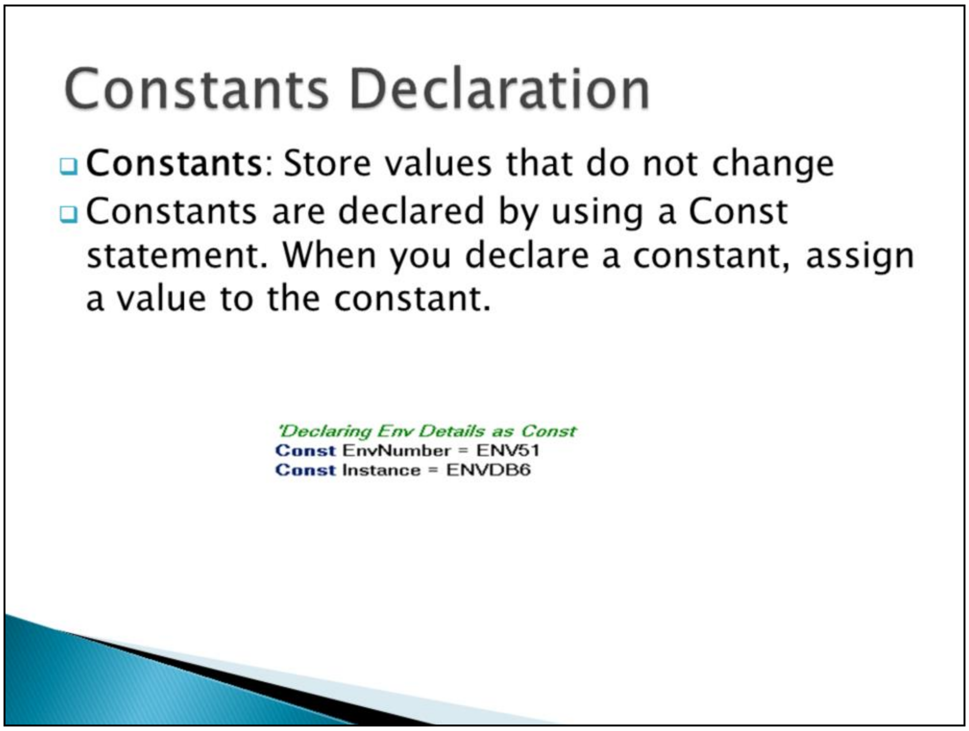
When multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

The string concatenation operator (**&**) is not an arithmetic operator, but in precedence it does fall after all arithmetic operators and before all comparison operators. The **Is** operator is an object reference comparison operator. It does not compare objects or their values; it checks only to determine if two object references refer to the same object.



Types Of VBScript Statements

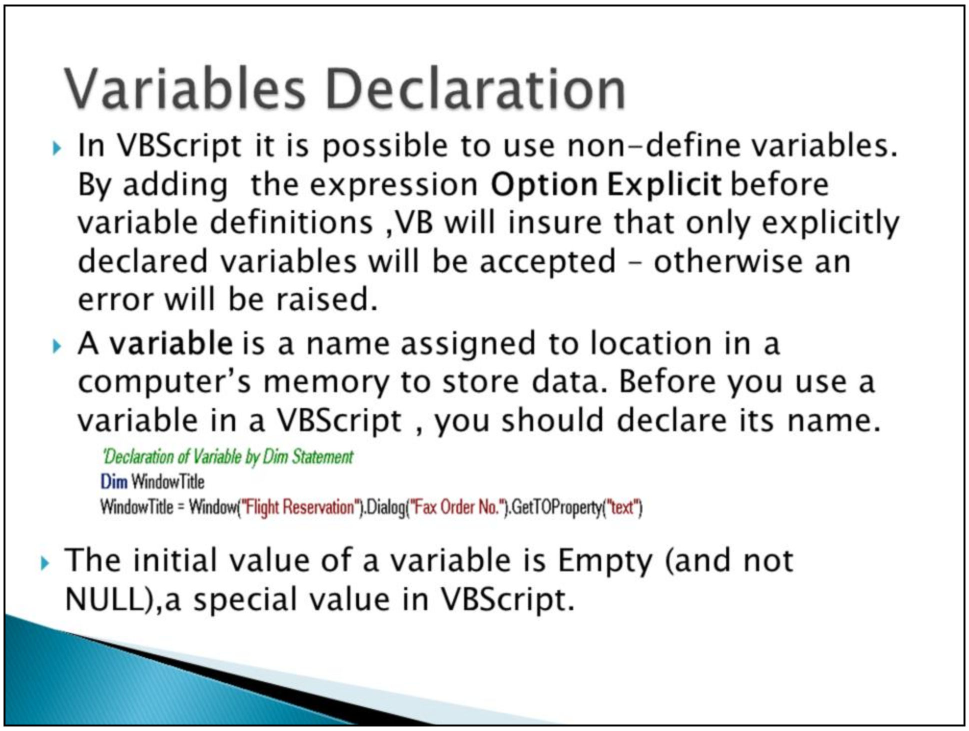
- A statement is a line of instruction in a program.
  A statement also refers to the keywords that designate the type of instruction to be performed.
- Commonly used VBScript statements include statements to:
  - Declare constants and variables.
  - Add comments to scripts.
  - Assign variables to point to objects.
  - Define conditional logic in scripts.
  - Define Looping Statement in scripts.
  - Create Class With Statement



Constants Declaration

- **Constants**: Store values that do not change
- Constants are declared by using a Const statement. When you declare a constant, assign a value to the constant.

```
'Declaring Env Details as Const
Const EnvNumber = ENV51
Const Instance = ENVDB6
```

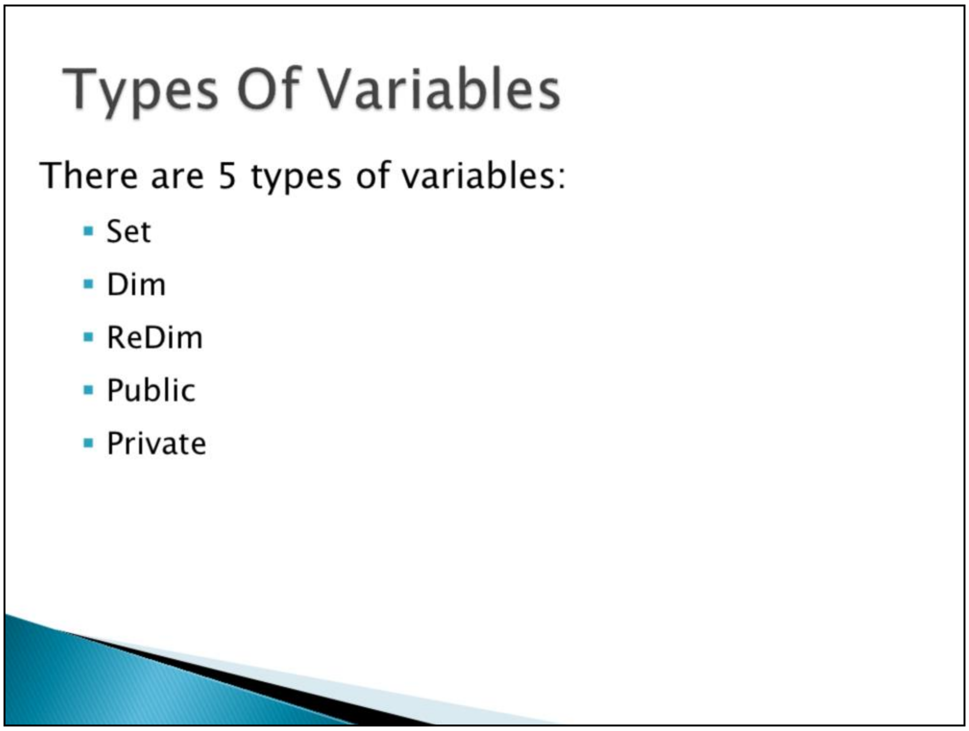


Variables Declaration

- In VBScript it is possible to use non-define variables. By adding the expression **Option Explicit** before variable definitions ,VB will insure that only explicitly declared variables will be accepted – otherwise an error will be raised.
- A **variable** is a name assigned to location in a computer's memory to store data. Before you use a variable in a VBScript , you should declare its name.

```
'Declaration of Variable by Dim Statement
Dim WindowTitle
WindowTitle = Window("Flight Reservation").Dialog("Fax Order No.").GetTOProperty("text")
```

- The initial value of a variable is Empty (and not NULL),a special value in VBScript.

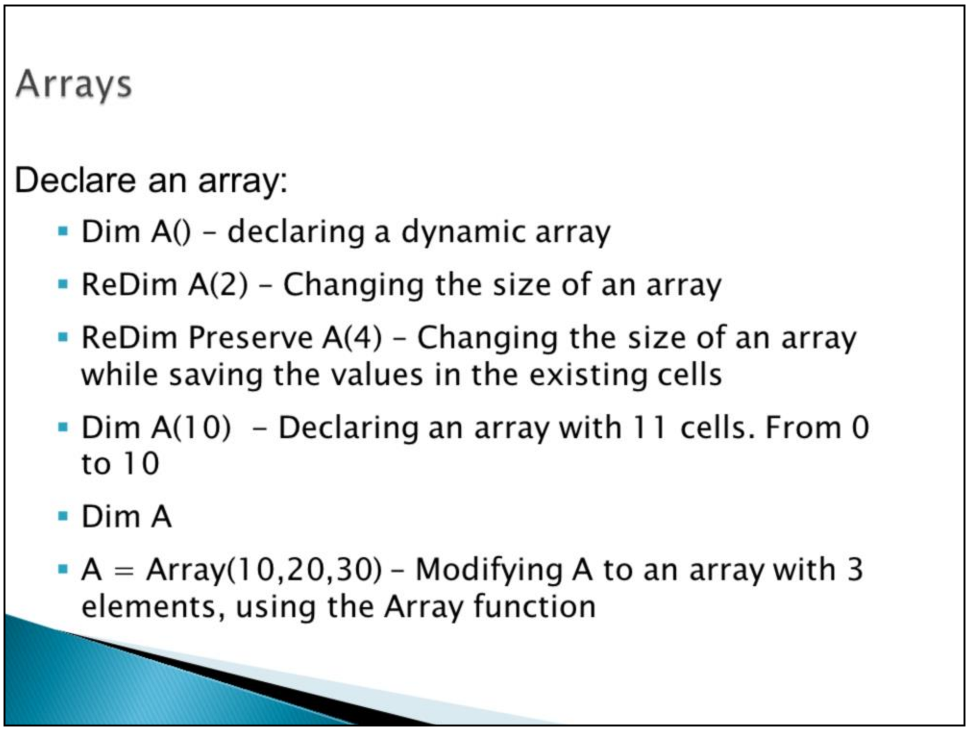


Types Of Variables

There are 5 types of variables:

- Set
- Dim
- ReDim
- Public
- Private

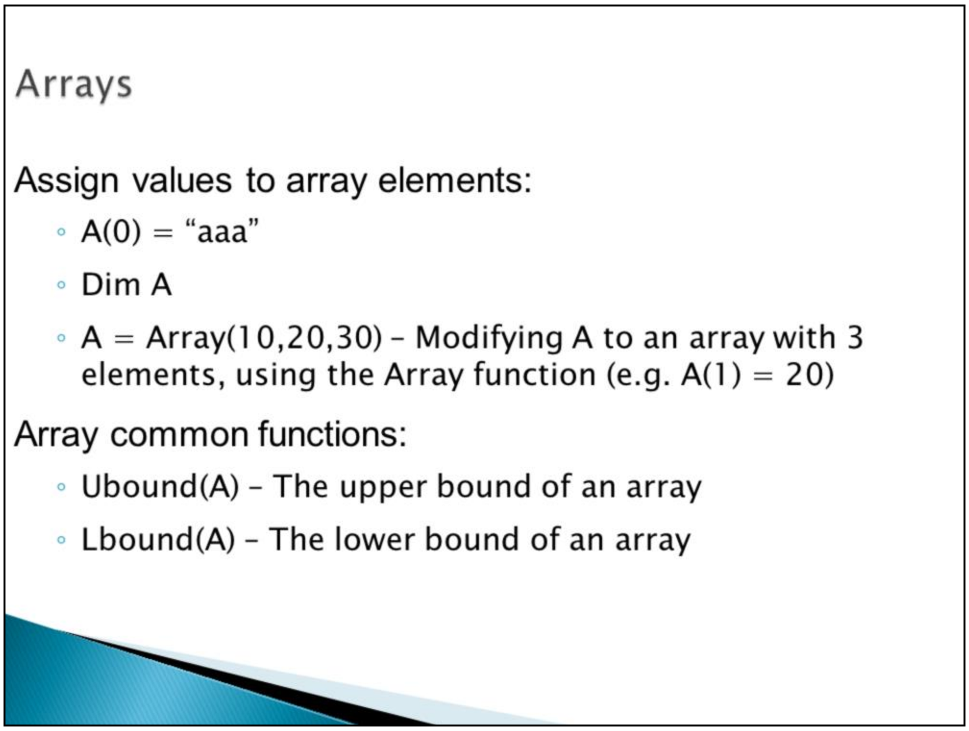
1. Set - Assigns an object reference to an object. When we set to object the **Nothing** Keyword it disassociates a variable from the object.
2. Dim - Declares variables and allocates storage space.
3. ReDim - Resize a dynamic array that has already been formally declared using a **Private**, **Public**, or **Dim** statement with empty parentheses .
4. Public - Declares public variables and allocates storage space.
5. Private - Declares private variables and allocates storage space



Arrays

Declare an array:

- Dim A() – declaring a dynamic array
- ReDim A(2) – Changing the size of an array
- ReDim Preserve A(4) – Changing the size of an array while saving the values in the existing cells
- Dim A(10) – Declaring an array with 11 cells. From 0 to 10
- Dim A
- A = Array(10,20,30) – Modifying A to an array with 3 elements, using the Array function



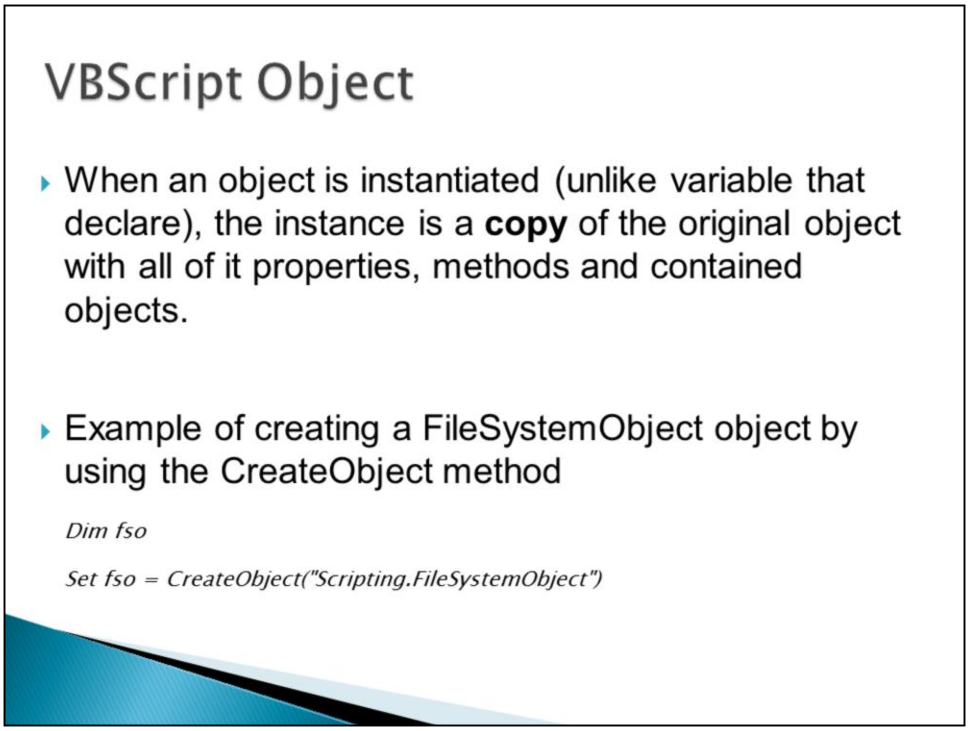
Arrays

Assign values to array elements:

- A(0) = “aaa”
- Dim A
- A = Array(10,20,30) – Modifying A to an array with 3 elements, using the Array function (e.g. A(1) = 20)

Array common functions:

- Ubound(A) – The upper bound of an array
- Lbound(A) – The lower bound of an array



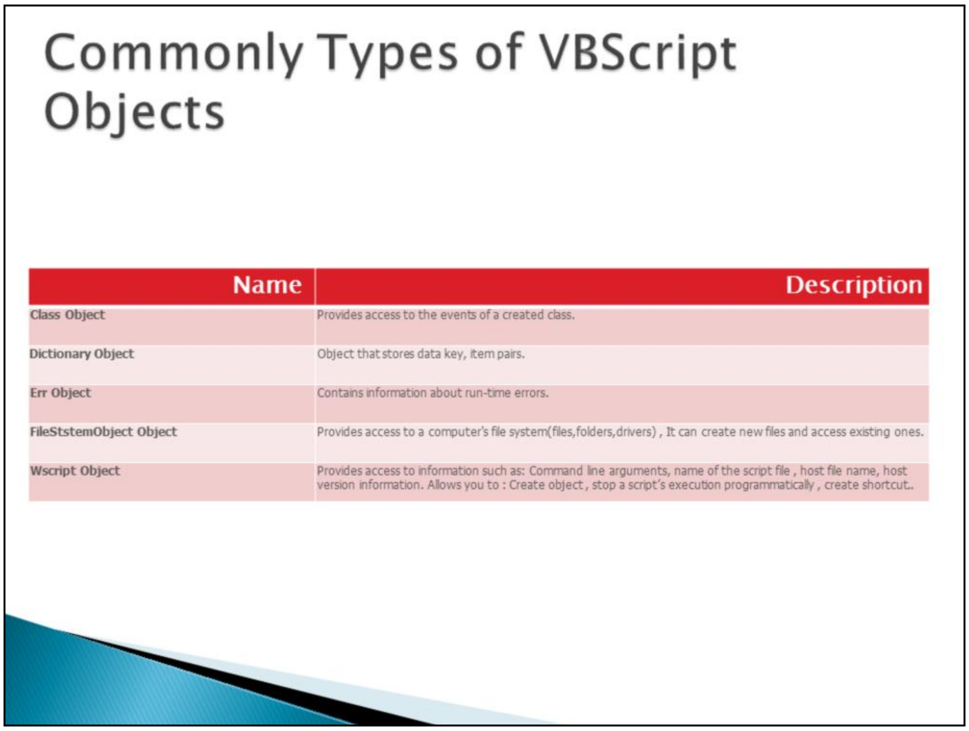
VBScript Object

- When an object is instantiated (unlike variable that declare), the instance is a **copy** of the original object with all of it properties, methods and contained objects.

- Example of creating a FileSystemObject object by using the CreateObject method

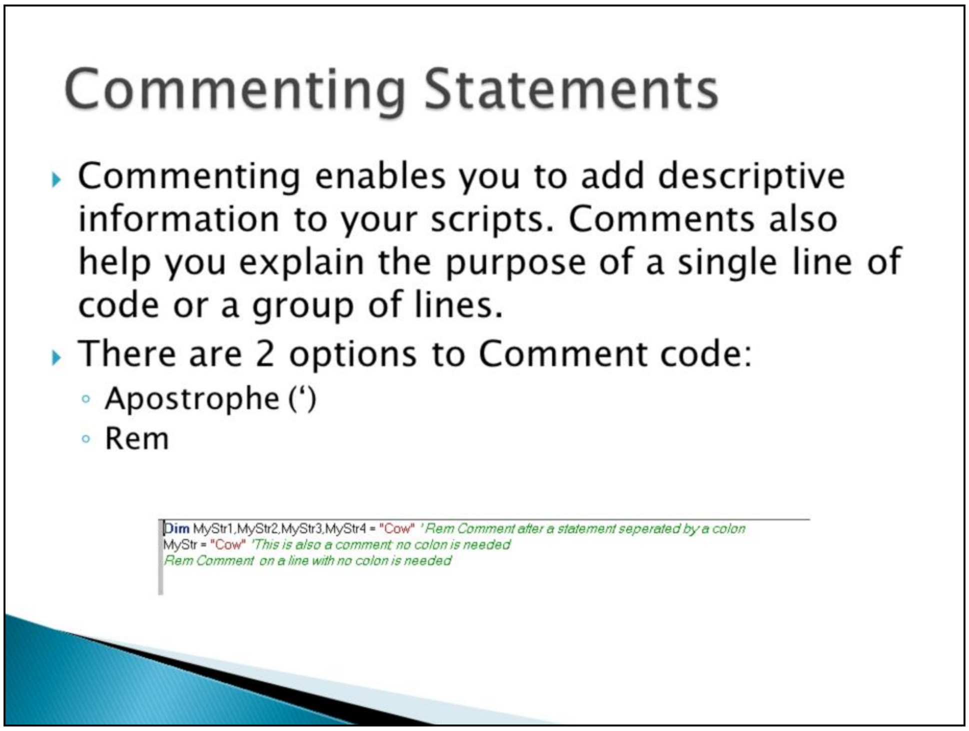
```
Dim fso

Set fso = CreateObject("Scripting.FileSystemObject")
```



Commonly Types of VBScript Objects

Name	Description
Class Object	Provides access to the events of a created class.
Dictionary Object	Object that stores data key, item pairs.
Err Object	Contains information about run-time errors.
FileStstemObject Object	Provides access to a computer's file system(files,folders,drivers) , It can create new files and access existing ones.
Wscript Object	Provides access to information such as: Command line arguments, name of the script file , host file name, host version information. Allows you to : Create object , stop a script's execution programmatically , create shortcut..

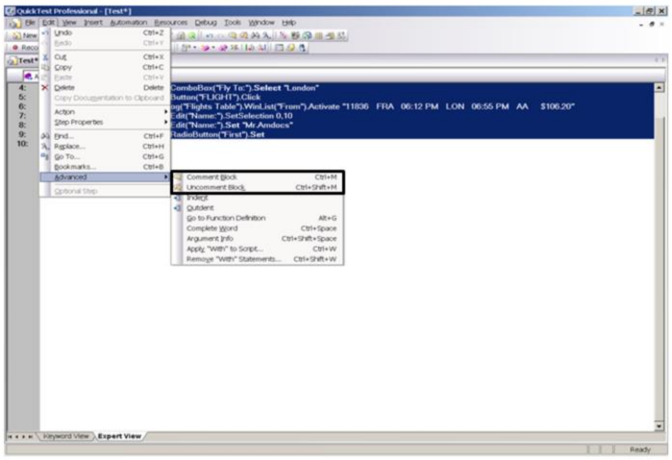


Commenting Statements

- Commenting enables you to add descriptive information to your scripts. Comments also help you explain the purpose of a single line of code or a group of lines.
- There are 2 options to Comment code:
  - Apostrophe (‘)
  - Rem

```
Dim MyStr1,MyStr2,MyStr3,MyStr4 = "Cow" 'Rem Comment after a statement seperated by a colon
MyStr = "Cow" 'This is also a comment; no colon is needed
Rem Comment  on a line with no colon is needed
```

Commenting Multiple Lines

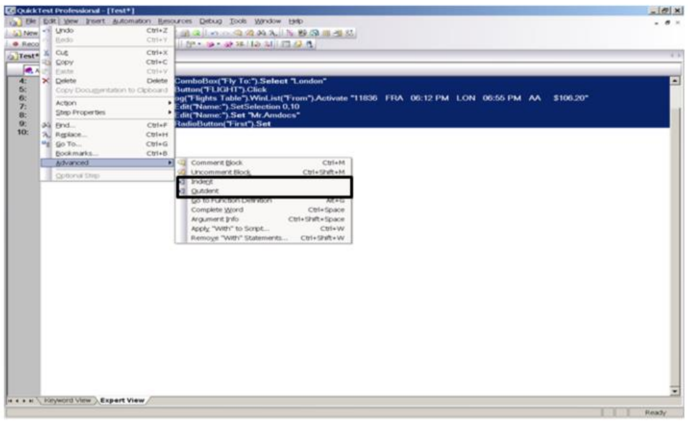


```
4:  ComboBox("Fly To:").Select "London"
5:  Button("FLIGHT").Click
6:  og("Flights Table").WinList("From").Activate "11836   FRA   06:12 PM   LON   06:55 PM   AA    $106.20"
7:  Edit("Name:").SetSelection 0,10
8:  Edit("Name:").Set "Mr.Amdocs"
9:  RadioButton("First").Set
10:
```

Comment Block - Comments out the current row, or selected rows.

Uncomment Block - Removes the comment formatting from the current or selected rows.

Indent And Outdent Feature

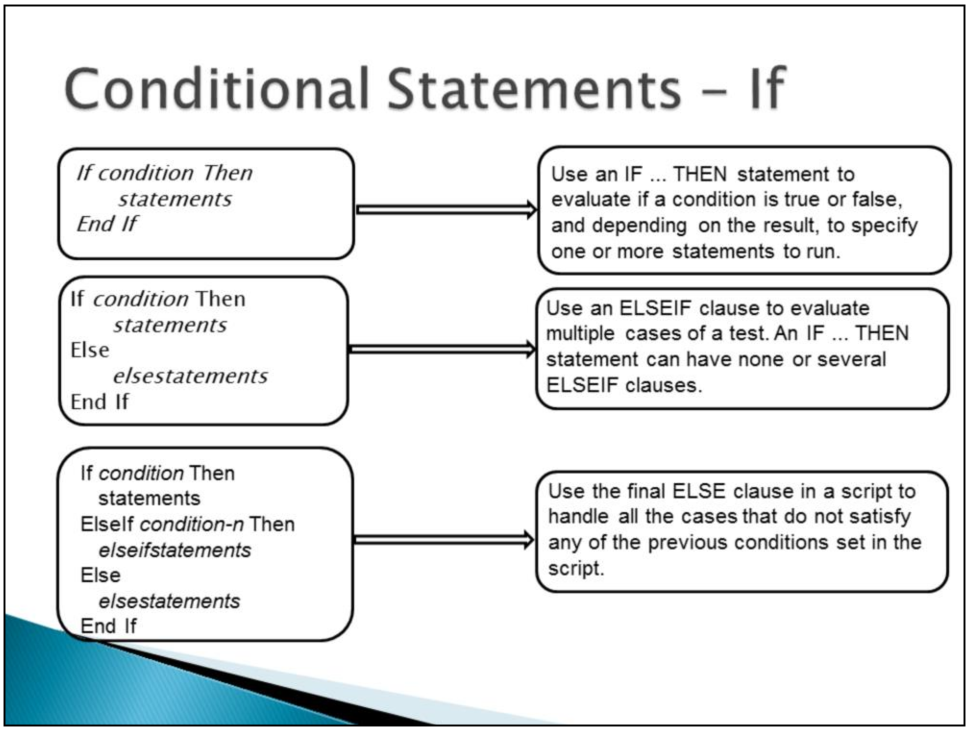

QuickTest Professional - [Test*]
File Edit View Insert Automation Resources Debug Tools Window Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Copy Documentation to Clipboard
Action
Step Properties
Find... Ctrl+F
Replace... Ctrl+H
Go To... Ctrl+G
Bookmarks... Ctrl+B
Advanced
Optional Step
Comment Block Ctrl+M
Uncomment Block Ctrl+Shift+M
Indent
Outdent
Go to Function Definition Alt+G
Complete Word Ctrl+Space
Argument Info Ctrl+Shift+Space
Apply "With" to Script... Ctrl+W
Remove "With" Statements... Ctrl+Shift+W
Keyword View
Expert View
Ready

```
ComboBox("Fly To:").Select "London"
Button("FLIGHT").Click
og("Flights Table").WinList("From").Activate "11836   FRA   06:12 PM   LON   06:55 PM   AA   $106.20"
Edit("Name:").SetSelection 0,10
Edit("Name:").Set "Mr.Amdocs"
RadioButton("First").Set
```

Indent - Indents the step according to the tab spacing defined in the Editor Options dialog box.

Outdent - Outdents the step (reduces the indentation) according to the tab spacing defined in the Editor Options dialog box.

Those features make your test actions more readable and easier to maintain.



Conditional Statements – If

```
If condition Then
    statements
End If
```

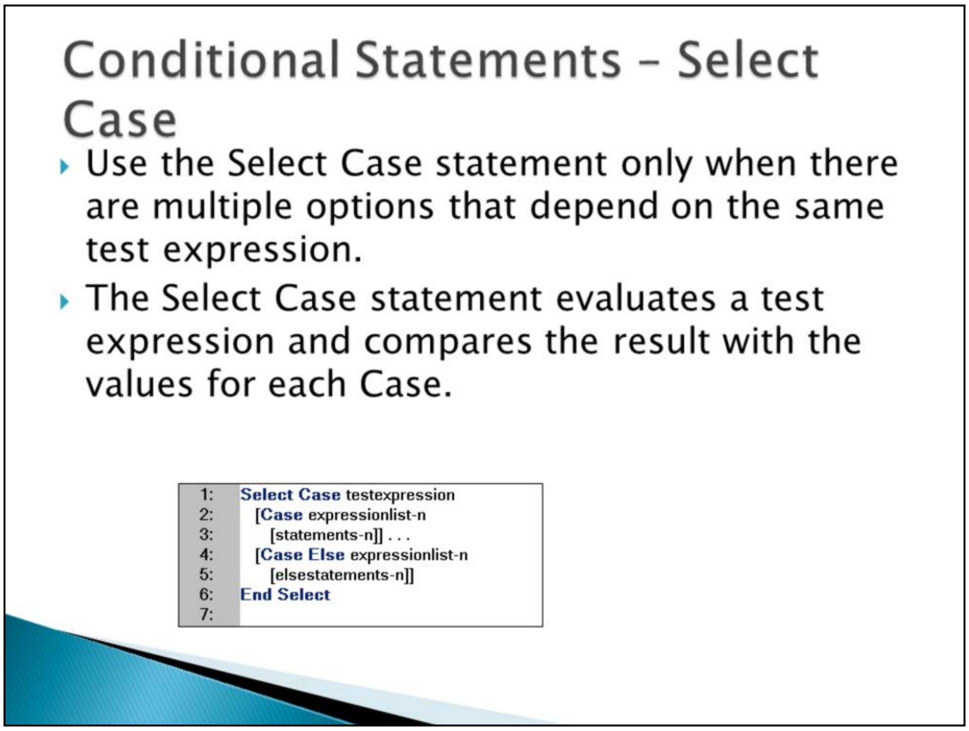
Use an IF ... THEN statement to evaluate if a condition is true or false, and depending on the result, to specify one or more statements to run.

```
If condition Then
    statements
Else
    elsestatements
End If
```

Use an ELSEIF clause to evaluate multiple cases of a test. An IF ... THEN statement can have none or several ELSEIF clauses.

```
If condition Then
  statements
ElseIf condition-n Then
  elseifstatements
Else
  elsestatements
End If
```

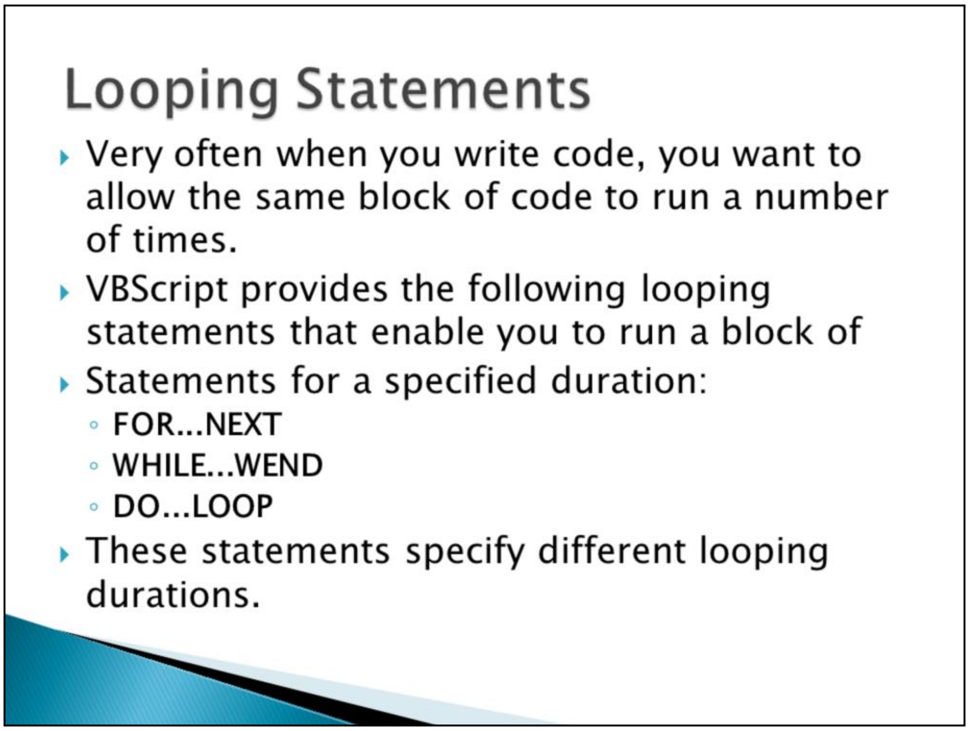
Use the final ELSE clause in a script to handle all the cases that do not satisfy any of the previous conditions set in the script.



Conditional Statements – Select Case

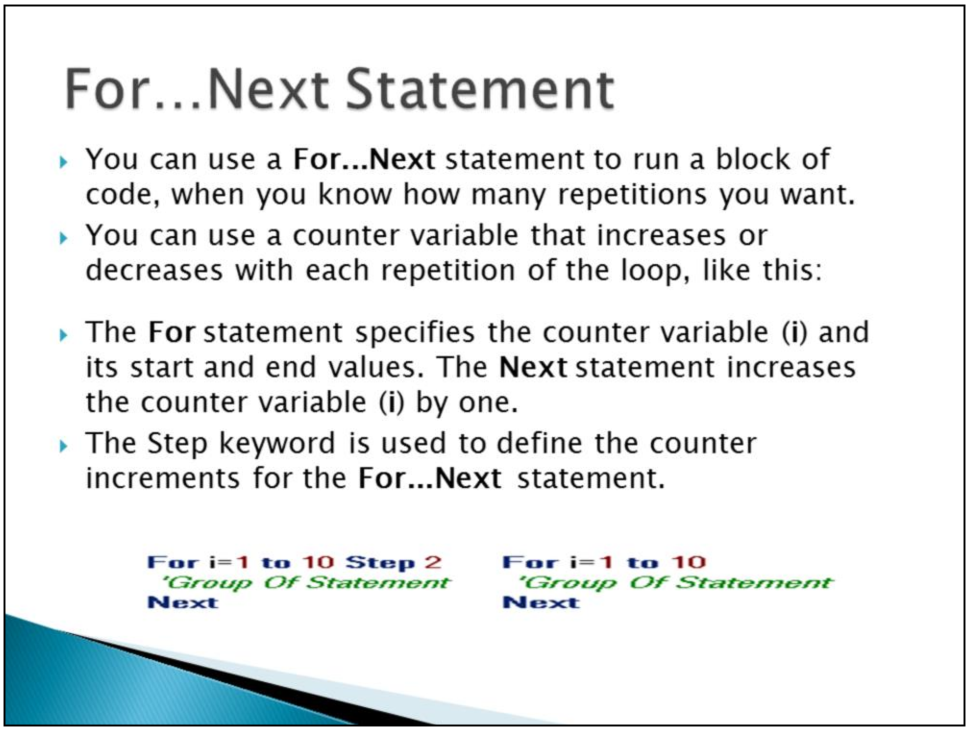
- Use the Select Case statement only when there are multiple options that depend on the same test expression.
- The Select Case statement evaluates a test expression and compares the result with the values for each Case.

```
Select Case testexpression
  [Case expressionlist-n
    [statements-n]] . . .
  [Case Else expressionlist-n
    [elsestatements-n]]
End Select
```



Looping Statements

- Very often when you write code, you want to allow the same block of code to run a number of times.
- VBScript provides the following looping statements that enable you to run a block of
- Statements for a specified duration:
  - FOR...NEXT
  - WHILE...WEND
  - DO...LOOP
- These statements specify different looping durations.

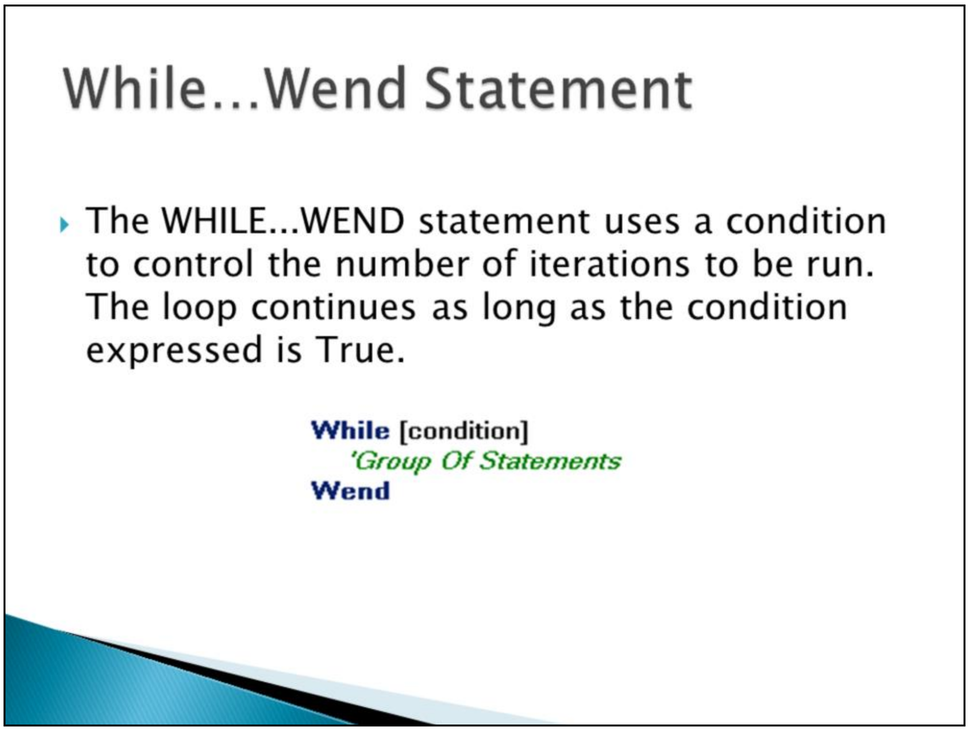


For…Next Statement

- You can use a **For...Next** statement to run a block of code, when you know how many repetitions you want.
- You can use a counter variable that increases or decreases with each repetition of the loop, like this:
- The **For** statement specifies the counter variable (**i**) and its start and end values. The **Next** statement increases the counter variable (**i**) by one.
- The Step keyword is used to define the counter increments for the **For...Next** statement.

```
For i=1 to 10 Step 2
 'Group Of Statement
Next
```

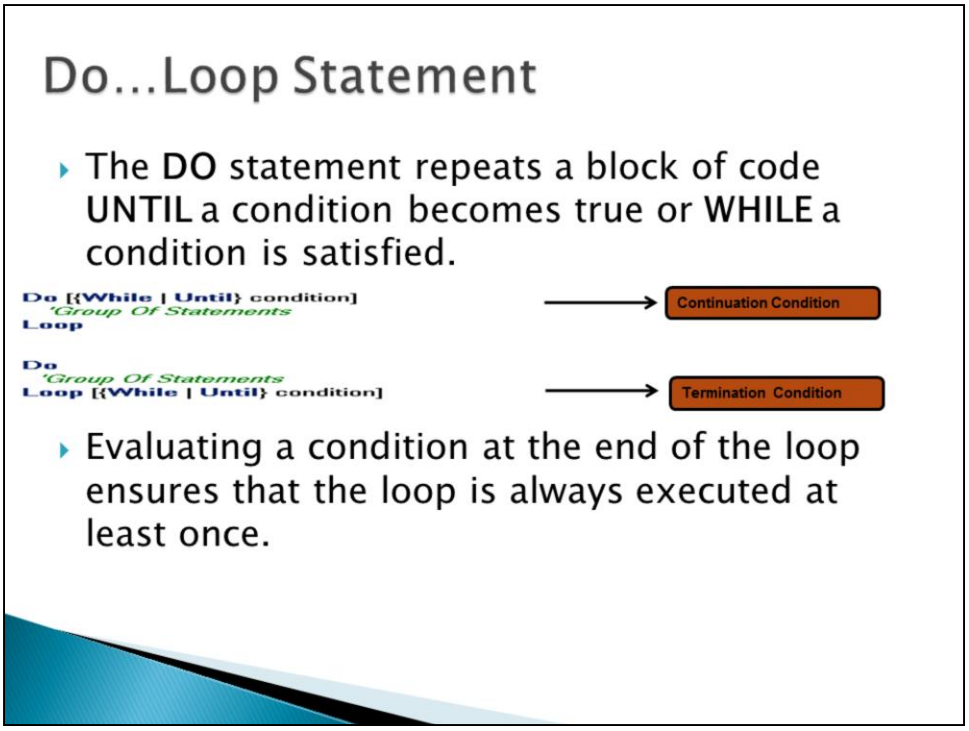
```
For i=1 to 10
 'Group Of Statement
Next
```



While…Wend Statement

- The WHILE...WEND statement uses a condition to control the number of iterations to be run. The loop continues as long as the condition expressed is True.

```
While [condition]
    'Group Of Statements
Wend
```



Do…Loop Statement

- The **DO** statement repeats a block of code **UNTIL** a condition becomes true or **WHILE** a condition is satisfied.

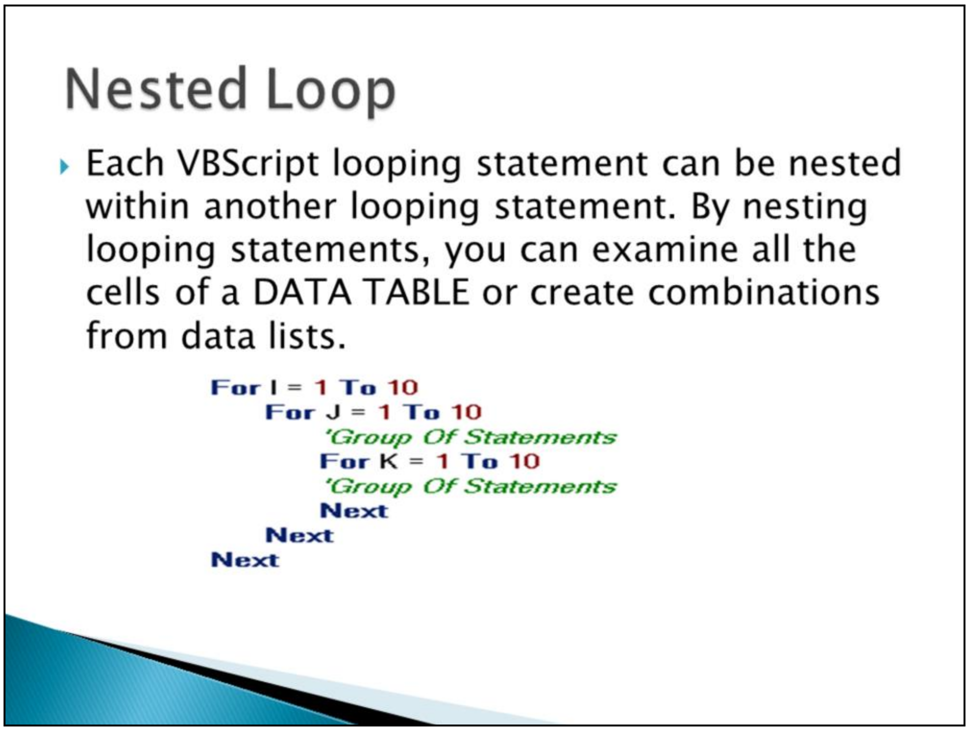
```
Do [{While | Until} condition]
   'Group Of Statements
Loop
```

$\longrightarrow$ Continuation Condition

```
Do
  'Group Of Statements
Loop [{While | Until} condition]
```

$\longrightarrow$ Termination Condition

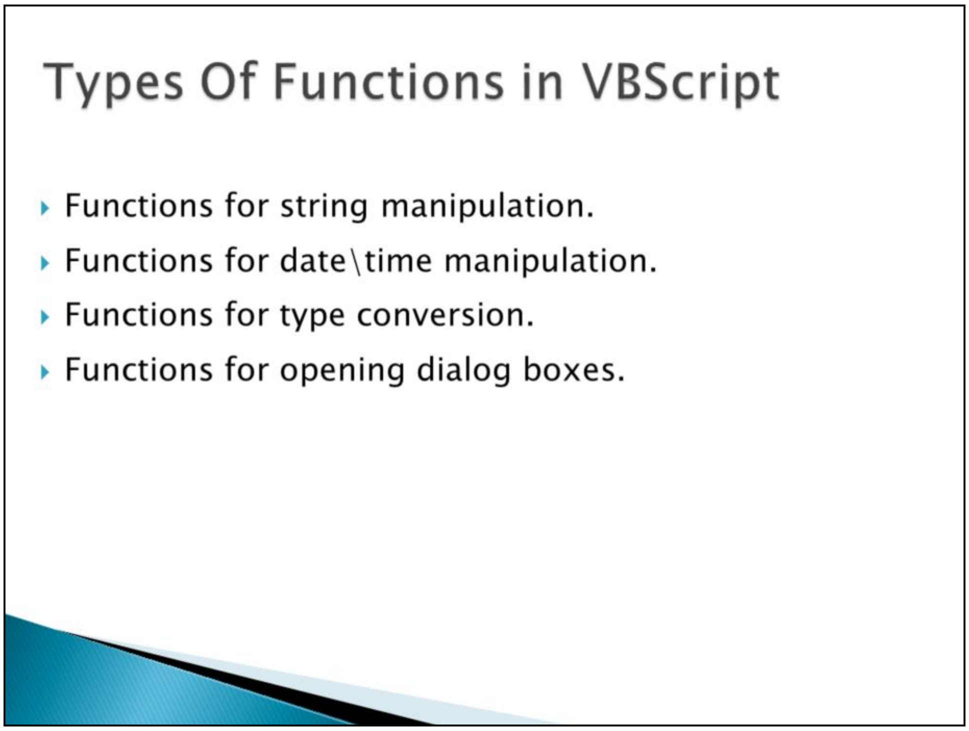
- Evaluating a condition at the end of the loop ensures that the loop is always executed at least once.



Nested Loop

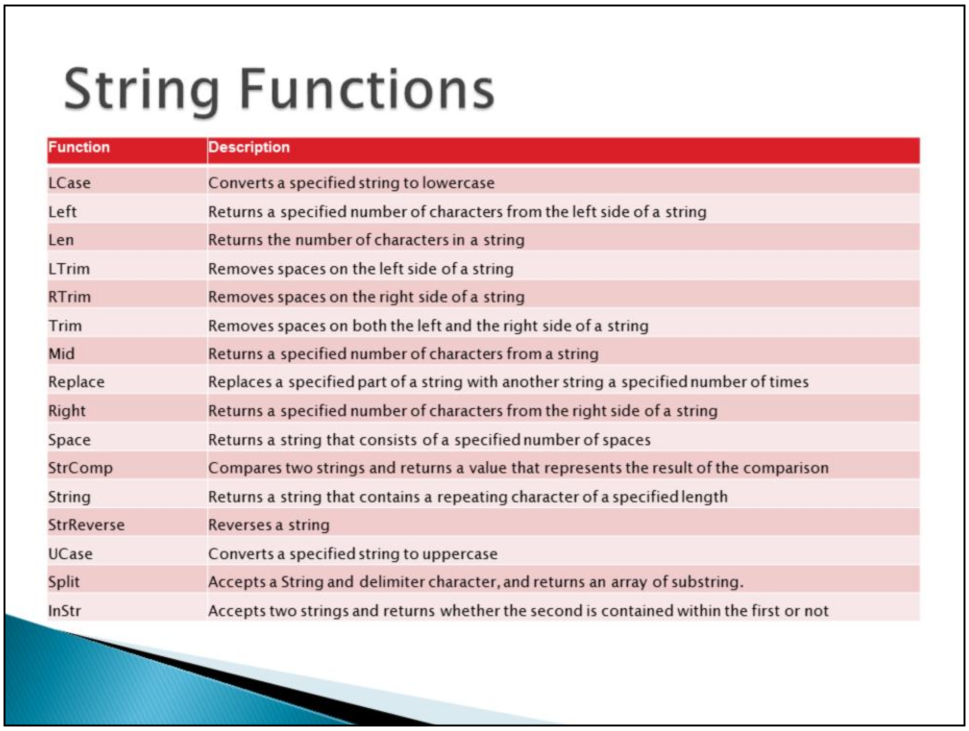
- Each VBScript looping statement can be nested within another looping statement. By nesting looping statements, you can examine all the cells of a DATA TABLE or create combinations from data lists.

```
For I = 1 To 10
    For J = 1 To 10
        'Group Of Statements
        For K = 1 To 10
        'Group Of Statements
        Next
    Next
Next
```



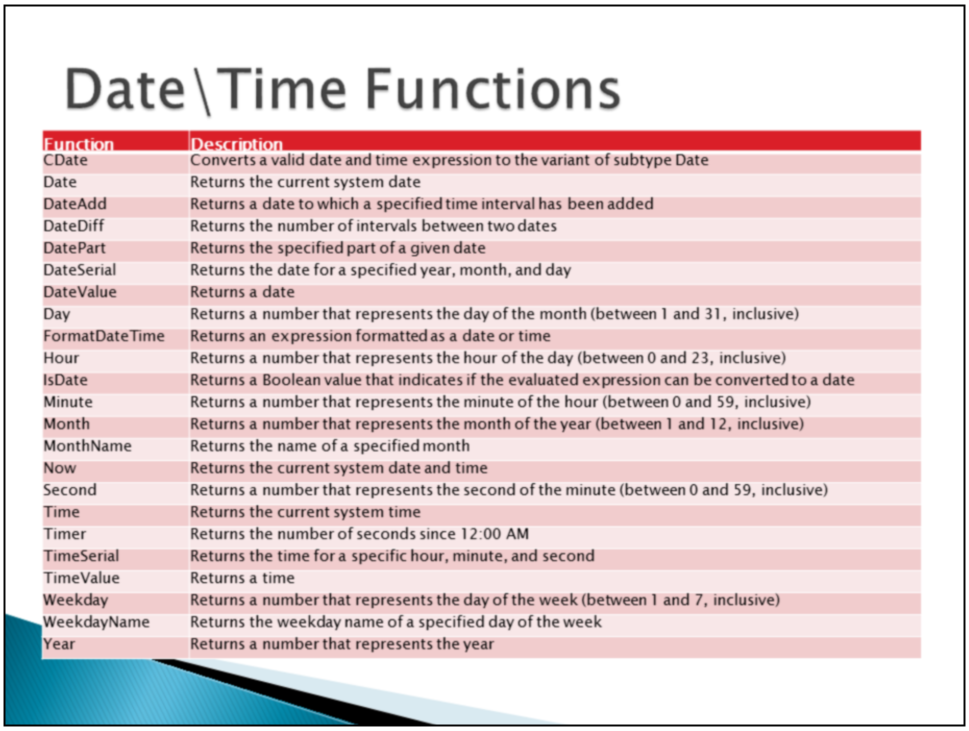
Types Of Functions in VBScript

- Functions for string manipulation.
- Functions for date\time manipulation.
- Functions for type conversion.
- Functions for opening dialog boxes.



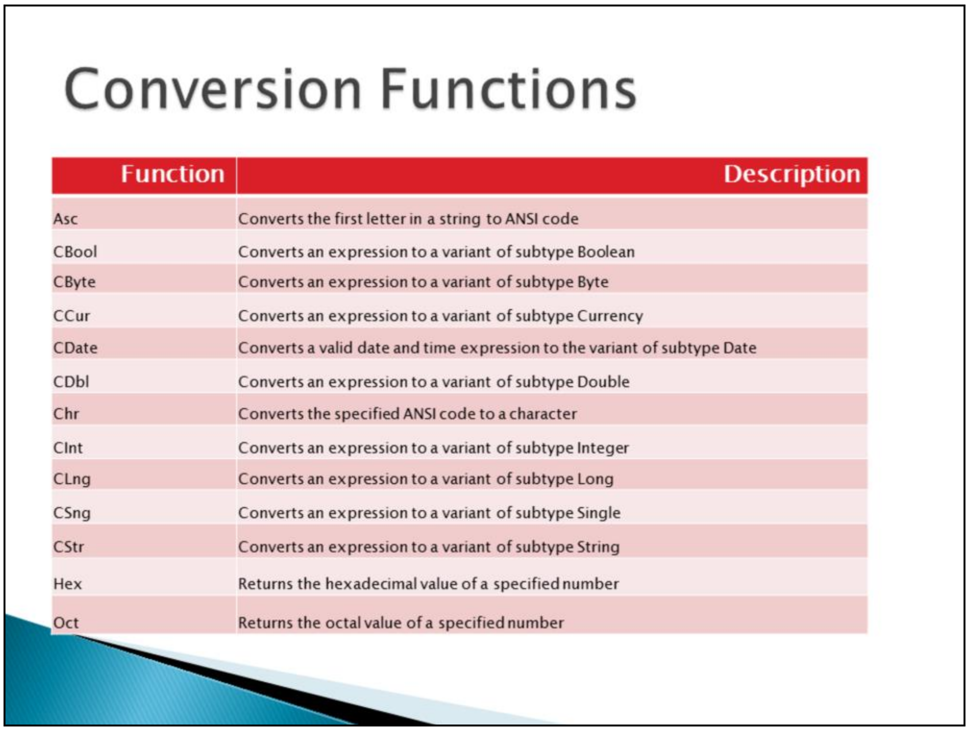
String Functions

Function	Description
LCase	Converts a specified string to lowercase
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
LTrim	Removes spaces on the left side of a string
RTrim	Removes spaces on the right side of a string
Trim	Removes spaces on both the left and the right side of a string
Mid	Returns a specified number of characters from a string
Replace	Replaces a specified part of a string with another string a specified number of times
Right	Returns a specified number of characters from the right side of a string
Space	Returns a string that consists of a specified number of spaces
StrComp	Compares two strings and returns a value that represents the result of the comparison
String	Returns a string that contains a repeating character of a specified length
StrReverse	Reverses a string
UCase	Converts a specified string to uppercase
Split	Accepts a String and delimiter character, and returns an array of substring.
InStr	Accepts two strings and returns whether the second is contained within the first or not



Date\Time Functions

Function	Description
CDate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a given date
DateSerial	Returns the date for a specified year, month, and day
DateValue	Returns a date
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
FormatDateTime	Returns an expression formatted as a date or time
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
TimeSerial	Returns the time for a specific hour, minute, and second
TimeValue	Returns a time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)
WeekdayName	Returns the weekday name of a specified day of the week
Year	Returns a number that represents the year



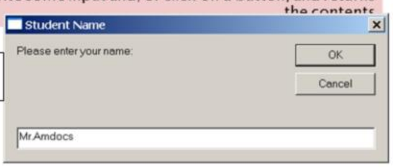
Conversion Functions

Function	Description
Asc	Converts the first letter in a string to ANSI code
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CCur	Converts an expression to a variant of subtype Currency
CDate	Converts a valid date and time expression to the variant of subtype Date
CDbl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
CInt	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CSng	Converts an expression to a variant of subtype Single
CStr	Converts an expression to a variant of subtype String
Hex	Returns the hexadecimal value of a specified number
Oct	Returns the octal value of a specified number

Opening Dialog Box Functions

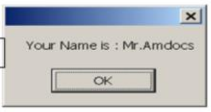
Function	Description
InputBox	Displays a dialog box, where the user can write some input and/or click on a button, and returns the contents

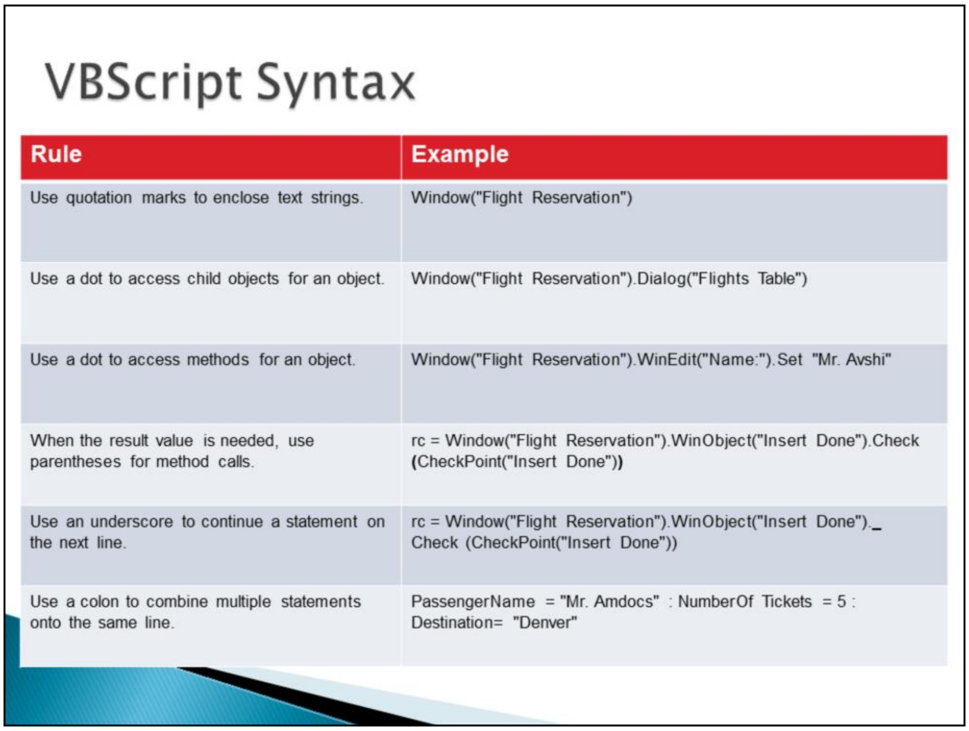
```
Dim Input
Input = InputBox("Please enter your name:" , "Student Name")
```



Function	Description
MsgBox	Displays a message box, waits for the user to click a button, and returns a value that indicates which button the user clicked

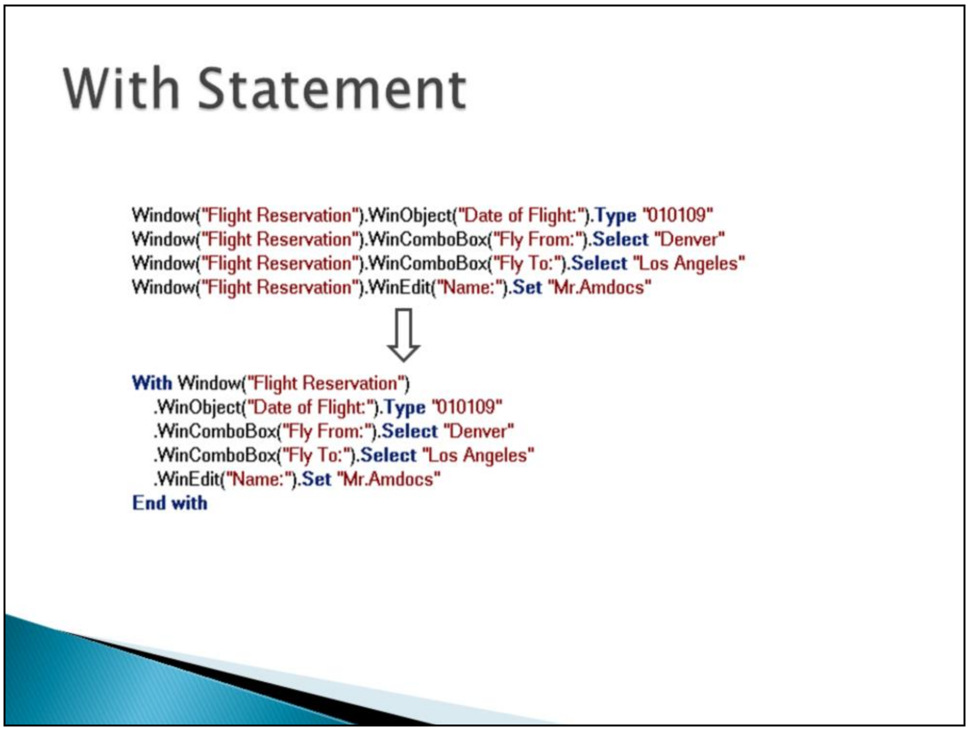
```
MsgBox "Your Name is : " & Input,vbOKOnly
```





VBScript Syntax

Rule	Example
Use quotation marks to enclose text strings.	Window("Flight Reservation")
Use a dot to access child objects for an object.	Window("Flight Reservation").Dialog("Flights Table")
Use a dot to access methods for an object.	Window("Flight Reservation").WinEdit("Name:").Set "Mr. Avshi"
When the result value is needed, use parentheses for method calls.	rc = Window("Flight Reservation").WinObject("Insert Done").Check **(**CheckPoint("Insert Done")**)**
Use an underscore to continue a statement on the next line.	rc = Window("Flight Reservation").WinObject("Insert Done").**_** Check (CheckPoint("Insert Done"))
Use a colon to combine multiple statements onto the same line.	PassengerName = "Mr. Amdocs" : NumberOf Tickets = 5 : Destination= "Denver"



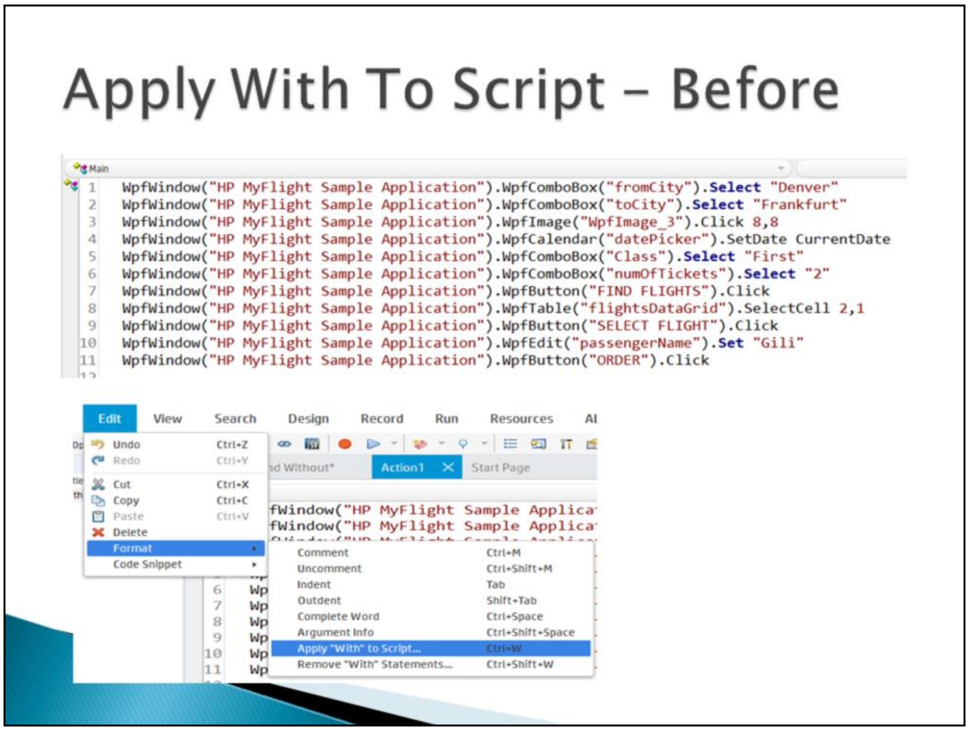
With Statement

```
Window("Flight Reservation").WinObject("Date of Flight:").Type "010109"
Window("Flight Reservation").WinComboBox("Fly From:").Select "Denver"
Window("Flight Reservation").WinComboBox("Fly To:").Select "Los Angeles"
Window("Flight Reservation").WinEdit("Name:").Set "Mr.Amdocs"
```

$\Downarrow$

```
With Window("Flight Reservation")
   .WinObject("Date of Flight:").Type "010109"
   .WinComboBox("Fly From:").Select "Denver"
   .WinComboBox("Fly To:").Select "Los Angeles"
   .WinEdit("Name:").Set "Mr.Amdocs"
End with
```

The **With** statement allows you to perform a series of statements on a specified object without including the name of the object.



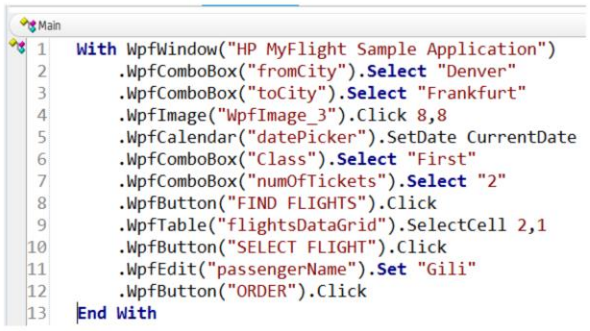
Apply With To Script – Before

```
WpfWindow("HP MyFlight Sample Application").WpfComboBox("fromCity").Select "Denver"
WpfWindow("HP MyFlight Sample Application").WpfComboBox("toCity").Select "Frankfurt"
WpfWindow("HP MyFlight Sample Application").WpfImage("WpfImage_3").Click 8,8
WpfWindow("HP MyFlight Sample Application").WpfCalendar("datePicker").SetDate CurrentDate
WpfWindow("HP MyFlight Sample Application").WpfComboBox("Class").Select "First"
WpfWindow("HP MyFlight Sample Application").WpfComboBox("numOfTickets").Select "2"
WpfWindow("HP MyFlight Sample Application").WpfButton("FIND FLIGHTS").Click
WpfWindow("HP MyFlight Sample Application").WpfTable("flightsDataGrid").SelectCell 2,1
WpfWindow("HP MyFlight Sample Application").WpfButton("SELECT FLIGHT").Click
WpfWindow("HP MyFlight Sample Application").WpfEdit("passengerName").Set "Gili"
WpfWindow("HP MyFlight Sample Application").WpfButton("ORDER").Click
```

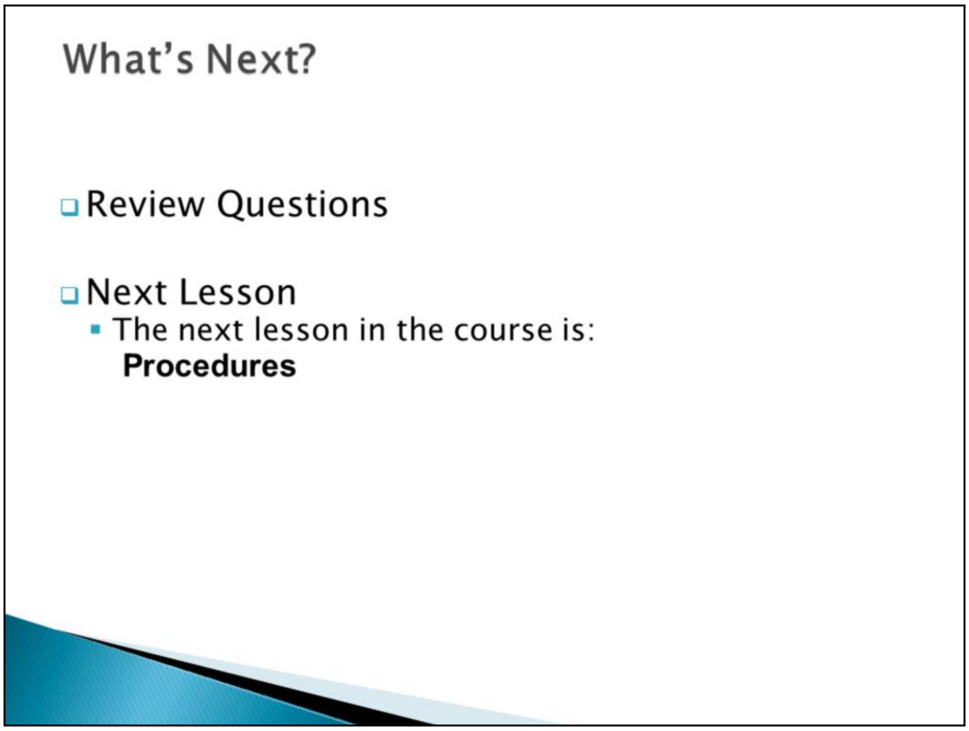
To convert a script so that sequential steps related to the same object are grouped together by using a With statement, from UFT menu bar, select **EDIT→FORMAT→APPLY “WITH” TO SCRIPT.**

To return the script to the previous format, from the UFT menu bar, select **EDIT→FORMAT→REMOVE “WITH” STATEMENTS.**

Apply With To Script – After



```
With WpfWindow("HP MyFlight Sample Application")
    .WpfComboBox("fromCity").Select "Denver"
    .WpfComboBox("toCity").Select "Frankfurt"
    .WpfImage("WpfImage_3").Click 8,8
    .WpfCalendar("datePicker").SetDate CurrentDate
    .WpfComboBox("Class").Select "First"
    .WpfComboBox("numOfTickets").Select "2"
    .WpfButton("FIND FLIGHTS").Click
    .WpfTable("flightsDataGrid").SelectCell 2,1
    .WpfButton("SELECT FLIGHT").Click
    .WpfEdit("passengerName").Set "Gili"
    .WpfButton("ORDER").Click
End With
```



What's Next?

- Review Questions
- Next Lesson
  - The next lesson in the course is: **Procedures**

End of Lesson