CDBL( ) – CONVERT TO DOUBLE

- Value in a textbox control is of String type. In order to perform Airthmatic Operations, the string type needs to be converted into double typeThe cdbl function converts an expression to type Double.

Syntax:

CDbl(expression)

Example:

```
txtSum.Text = CDbl(txtFirst.Text) + CDbl(txtSecond.Text)
```

CSTR( ) – CONVERT TO STRING

The CStr function converts an expression into a String data type.

The syntax of the function is:

CStr(Expression)

Example:

```
' Convert Integer, Boolean and Date expressions into Strings
Dim str1 As String
Dim str2 As String
Dim str3 As String
str1 = CStr( 10 )
' str1 is now equal to the string "10"
str2 = CStr( True )
'str2 is now equal to the string "True"
str3 = CStr( #1/1/2016# )
' str3 is now equal to the string "1/1/2016"
```

NOW( )

- The Now Function returns the current date and time. The function takes no parameters and therefore, its syntax is : now()

Example:

```
' Store the current date and time in the variable dateTime
Dim dateTime As Date
dateTime = Now()
```

SPACE ( )

The Space function creates a String consisting of a specified number of spaces.

syntax :

Space(Number)

Where the Number argument is the number of spaces making up the returned String.

Example( ):

```
    ' Example 1 - Create a String containing 5 spaces.
Dim str1 As String
str1 = Space( 5 )
    ' The variable str1 is now equal to "     " (five spaces).
    ' Example 2 - Create a String containing 0 spaces.
Dim str2 As String
str2 = Space( 0 )
    ' The variable str2 is now equal to "" (an empty string).
```

ISNUMERIC( )

The IsNumeric function returns a Boolean, indicating whether a supplied expression is interpreted as a numeric value.

syntax :

IsNumeric(Expression)

Where the supplied Expression is the expression that you want to test

Example:

```
' Test if five different expressions are numeric.
Dim isNum1 As Boolean
Dim isNum2 As Boolean
isNum1 = IsNumeric( 5 )
' The variable isNum1 is now equal to True.
isNum5 = IsNumeric( "A55" )
' The variable isNum2 is now equal to False
```

IIF( )

- The IIf function evaluates an expression and returns one of two values, depending on whether the expression evaluates to True or False.

syntax :

IIf(Expression, TruePart, FalsePart)

Example:

```
' Test if a Supplied Integer is Positive or Negative.
Dim testVal As Integer
Dim sign1 As String
Dim sign2 As String
' call to IIf function. The test value is negative:
testVal = -2
sign1 = IIf( testVal < 0, "negative", "positive" )
' sign1 is now equal to "negative".
```

LTRIM( )

The LTrim function removes the leading spaces from a supplied text string.

syntax :

LTrim(String)

Where the String argument is the text string that you want to remove the leading spaces from.

Example:

```
' Remove the leading spaces from text strings.
Dim address As String
address = "     21 Smithson Street"
 ' has several leading spaces
address = LTrim( address )
' After running the above code,
' address = "21 Smithson Street"
```

VAL( )

- **The Val function converts a supplied string into a numeric value.**

syntax :

Val(String)

Where the String argument is the string that you want to convert into a number.

Example:

```
' Convert four strings into numeric values.
Dim num1 As integer
Dim num2 As Double
num1 = Val( "500" )
' num1 is now equal to 500.
num2 = Val( "+10.9" )
' num2 is now equal to 10.9.
```

UBOUND( )

- **The UBound function returns the highest subscript for a dimension of a supplied array.**

syntax :

UBound(ArrayName, [Dimension])

ArrayName	-	The array for which you want to find the highest subscript.
[Dimension]	-	An optional integer, specifying the dimension of the array, for which you require the highest subscript. If [Dimension] is omitted, it takes on the default value 1.

Example:

```
                ' Return the highest subscript of a one-dimensional array.
Dim prices(0 to 10) As Double
Dim pricesUB As Integer
pricesUB = UBound( prices )
                ' Now the integer pricesUB has the value 10.


                ' Return the highest subscripts of each dimension a 2-d array.
Dim costs(0 to 10, 0 to 100) As Double
Dim costsUB1 As Integer
Dim costsUB2 As Integer
costsUB1 = UBound( costs, 1 )
costsUB2 = UBound( costs, 2 )
                ' Now, costsUB1 = 10 and costsUB2 = 100.
```

LBOUND( )

- **The LBound function returns the Lowest subscript for a dimension of a supplied array.**

syntax :

UBound(ArrayName, [Dimension])

ArrayName	-	The array for which you want to find the Lowest subscript.
[Dimension]	-	An optional integer, specifying the dimension of the array, for which you require the Lowest subscript. If [Dimension] is omitted, it takes on the default value 1.

Example:

```
                ' Return the Lowest subscript of a one-dimensional array.
Dim prices(0 to 10) As Double
Dim pricesLB As Integer
 pricesLB = LBound( prices )
                ' Now the integer pricesLB has the value 0.


                ' Return the Lowest subscripts of each dimension a 2-d array.
Dim costs(0 to 10, 1 to 100) As Double
Dim costsLB1 As Integer
Dim costsLB2 As Integer
costsLB1 = LBound( costs, 1 )
costsLB2 = LBound( costs, 2 )
                ' Now, costsLB1 = 0 and costsLB2 = 1.
```