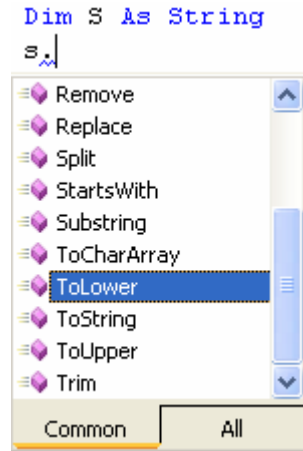


Strings

Many string functions have two forms. For instance, if a variable is declared as a string, typing a dot after the variable will cause a list of methods for the string class to appear.



The ToLower method returns the string in lower case letters. The code below will display “to be or not to be” as the text. The value of S is not changed. (The ToLower method only affects letters of the alphabet.)

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
    Dim S As String = "TO BE OR NOT TO BE"
    Me.Text = S.ToLower
End Sub 'Button1_Click
```

There is also a function LCase that can be used if the data is not stored as a string:

```
Me.Text = LCase("TO BE OR NOT TO BE")
```

(Note: Two functions that do the same thing may not have the same name. This is often done to make code backward compatible. Code written in an earlier version to the language will still work.)

For each example in the table below, the value of S is “**Visual Basic: Strings**”

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    Dim S As String
    S = "Visual Basic: Strings"
    Me.Text = S.____ 'try various methods
End Sub 'Form1_Load
```

Method in String Class	Example	Returns
Contains	S.Contains("B")	True
	S.Contains("b")	False <i>'this method is case sensitive</i>
EndsWith	S.EndsWith("ings")	True
	S.EndsWith("iNGs")	False
*S.EndsWith("iNGs", StringComparison.CurrentCultureIgnoreCase) returns True		
<i>All comparison methods allow this argument.</i>		

For each example in the table below, the value of S is **“Visual Basic: Strings”**

IndexOf	S.IndexOf("i")	1 'This is a zero based index
	S.IndexOf("w")	-1 'Not found
	S.IndexOf("i", 3)	10 'Start search in position 3
	S.IndexOf("a")	4
	S.IndexOf("a", 0, 3)	-1 'Start search in 0, look at 3 char. Only
Insert	S.Insert(5, "XX")	Visual Basic: Strings 'insert XX at pos. 5
LastIndexOf	S.LastIndexOf("i")	17 'Start search at end. This has same functionality as IndexOf
Length	S.Length()	21 'Counts everything in string, including spaces and punctuation.
Remove	S.Remove(3, 5)	Visual Basic: Strings 'Starts at pos. 3 and removes 5 letters. (Visual Basic: Strings)
Replace	S.Replace("s", "X")	Visual Basic: StringX 'Replace s with X
	S.Replace(" ", "")	Visual Basic: Strings 'Replace space with nothing: The 2 strings do not have to be the same length.
Split	S.Split(" ")	Splits a string in to an array. The result cannot be displayed in the text:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    Dim Words(), S As String
    S = "Visual Basic: Strings"
    Words = S.Split(" ") 'Split at each space
    Me.Text = Words(0) 'Displays the word Visual
End Sub 'Form1_Load
```

StartsWith	S.StartsWith("Vi")	True 'This has the same functionality as EndsWith
Substring	S.Substring(3, 5)	Visual Basic: Strings 'Starts at pos. 3 and returns 5 letters.
	S.Substring(3)	Visual Basic: Strings 'If no length is given it returns everything from that pos. on.
ToCharArray		Returns an array of char. The result cannot be displayed in the text:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
    S As String
    Dim Ch() As Char
    S = "Visual Basic: Strings"
    Ch = S.ToCharArray() 'Returns an array of characters
    Me.Text = Ch(0) 'Displays V
End Sub 'Form1_Load
```

ToLower	S.ToLower	visual basic: strings
ToUpper	S.ToUpper	VISUAL BASIC: STRINGS
Trim	S = S.Trim	'Removes blanks from the ends of a string.

There is also a ToString function that can be used with most objects. Me.Text = Me.ToString displays “strings.Form1, Text: Form1”. The statement Me.Button1.ToString displays System.Windows.Forms.Button, Text:Button1.

These functions can be used without a string variable.

Function/Syntax S = string, N =integer	Description	Example → Value Returned
Asc(S)	Converts 1 st character to ASCII	Asc("%") → 37
Chr(ascii#)	Converts ASCII to a character	Chr(37) → "%" Chr(241) → "ñ"
InStr(S1, S2) InStr(N, S1, S2)	Returns starting position of S2 within S1. Begins search at N if given. <i>Returns 0 if not found</i>	InStr("It is fun", " ") → 3 InStr(4, "It is fun", " ") → 6 InStr("It is now", "B") → 0
LCase(S)	converts string to lower case	LCase("ABC") → "abc"
Left(S, N)	returns leftmost N characters	Left("ABCDEF", 3) → "ABC"
Len(S)	returns length of string	Len("ABCDEF") → 6
Mid(S, F, L) Mid(S, F)	extracts a portion of a string: from F for a length of L. If L is omitted, takes everything from F to end.	Mid("ABCDEF", 3, 2) → "CD" Mid("ABCDEF", 3) → "CDEF"
Right(S, N)	returns rightmost N characters	Right("ABCDEF", 3) → "DEF"
Space(N)	Returns N spaces	Space(3) → " "
String(N, S)	Returns N of the 1 st character	String(3, "*") → "***" String(3, "abc") → "aaa"
LTrim(S)	Trims leading blanks	LTrim(" ABC ") → "ABC "
RTrim(S)	Trims trailing blanks	RTrim(" ABC ") → " ABC"
Trim(S)	Trims leading and trailing blanks	Trim(" ABC ") → "ABC"
UCase(S)	converts string to upper case	UCase("Hello") → "HELLO"

The Mid function can be used to extract each character one at a time: To test the sample code put a button on the form.

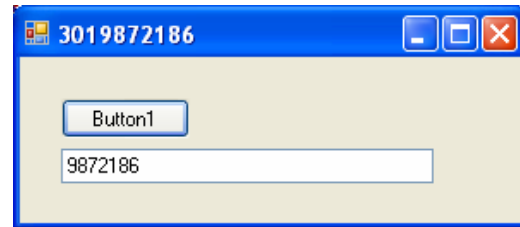
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
    Dim S As String
    Dim N As Integer
    Dim Ch As Char
    S = Me.TxtInput.Text
    Me.Text = ""
    For N = 1 To Len(S)
        Ch = Mid(S, N, 1) 'begin in position N for 1 char
        Me.Text = Me.Text & Ch
    Next N
End Sub 'Button1_Click
```

Experiment: Modify the procedure above to display each letter in uppercase with * between each letter: Example: When the user types "Hello" the caption is displayed as "H*E*L*L*O".

Phone Numbers

The string functions become more interesting when used in combination. Suppose you have a list of phone numbers. Some include the area code some do not. If the area code is missing, it is assumed to be local: 301 for instance. Furthermore some of the phone numbers have other characters than just digits. We would like to standardize the format of the numbers so that each number is 10 digits.

Original	Final
301-123-1284	3011231284
456-2345	3014562345
(703) 555-1212	7035551212
202.428.9187	2024289187
9872186	3019872186



This program has an text box and a button and simply displays the result in the text. This is a good way to test code. After testing, it can be made into a function that receives the number and returns the result.

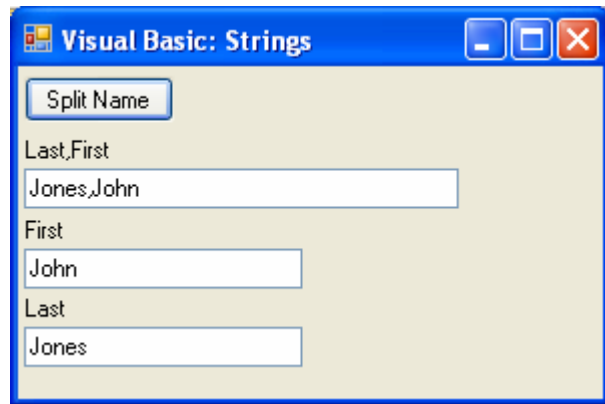
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click
    Dim Local As String = "301"
    Dim S As String
    Dim N As Integer
    Dim Ch As Char
    Dim Phone As String
    S = Me.TxtInput.Text
    Phone = ""
    For N = 1 To Len(S) 'look at each char, add to phone if digit
        Ch = Mid(S, N, 1) 'begin in position N for 1 char
        If Ch >= "0" And Ch <= "9" Then
            Phone = Phone & Ch
        End If
    Next N
    If Phone.Length = 7 Then
        Phone = Local & Phone
    End If
    If Phone.Length <> 10 Then
        Me.Text = "INVALID NUMBER"
    Else
        Me.Text = Phone
    End If
End Sub 'Button1_Click
```

Experiment: Make this into a function that receives the phone number and returns the standardized number.

Experiment: Use format to display a standardized phone number in the format (301) 257-1580.

Names

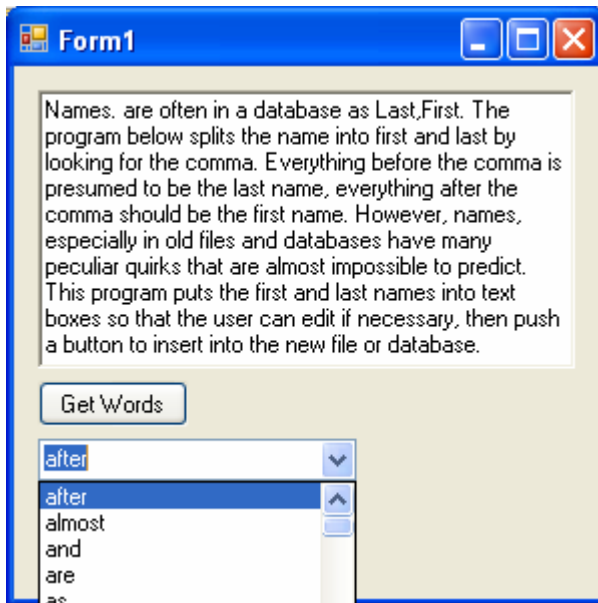
Names are often in a database as Last,First. The program below splits the name into first and last by looking for the comma. Everything before the comma is presumed to be the last name, everything after the comma should be the first name. However, names, especially in old files and databases have many peculiar quirks that are almost impossible to predict. This program puts the first and last names into text boxes so that the user can edit if necessary, then push a button to insert into the new file or database.



```
Private Sub BtnName_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles BtnName.Click
    Dim S As String
    Dim P As Integer
    S = Me.TxtInput.Text
    P = S.IndexOf(",")
    Me.TxtLast.Text = S.Substring(0, P) 'up to the comma
    Me.TxtFirst.Text = S.Substring(P + 1)
        'after the comma - don't include comma
End Sub 'BtnName_Click
```

Words in a Paragraph

Start a new program and name it Words. Add a Rich Text Box control named RchParagraph, a button BtnWords, and a Combo Box names CboWords. Set the sorted property of the combo box to true. Paste some text into the Richtext box.



The code is shown below:

```
Private Sub BtnWords_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles BtnWords.Click
    Dim Words(), Ch, S As String
    Dim Punc As String = ".,:;?!"
    Dim N, W, L, P As Integer
    S = Me.RchParagraph.Text
    Words = S.Split(" ") 'Split at each space
    N = Words.Length
    For W = 0 To N - 1 'each word
        For L = 0 To Punc.Length - 1 'each character in punc
            Ch = Punc.Substring(L, 1)
            Words(W) = Words(W).Replace(Ch, "") 'replace all punc. with nothing
        Next L 'next punctuation mark
        P = Me.CboWords.FindString(Words(W))
        If P = -1 Then 'no duplicates in list!
            Me.CboWords.Items.Add(Words(W))
        End If
    Next W 'next word
    Me.CboWords.SelectedIndex = 0 'display first word
End Sub 'BtnWords_Click
```

Experiment: Count how many times each word appears.