Mr. Giansante

Visual Basic

Text and Strings

August 2016

String Functions

Visual Basic offers many functions that can operate on string variables.

The examples on this page assume you have made the following declaration.

```
Dim myString As String
myString = "Programming is fun"
```

Length of a String

The Length property returns the number of characters in the string.

```
n = myString.Length()
```

In this case, the variable n would be set to 18 since there are 18 characters in "Programming is fun".
Note: spaces are counted as characters.

Changing Case

To change all the characters in a string to uppercase ...

```
myString.ToUpper()
```

To change all the characters in a string to lowercase ...

```
myString.ToLower()
```

Removing Spaces

To delete spaces at the beginning of a string ...

```
myString.TrimStart()
```

To delete spaces at the end of a string ...

```
myString.TrimEnd()
```

To delete spaces at the beginning and end of a string ...

```
myString.Trim()
```

Adding Spaces

You can use `PadLeft` and `PadRight` to add spaces (or any other character) to the start or end of a string.

```
myString = myString.PadLeft(___, " ")
```

Total Length of the new string.

Note: This is NOT how many characters you want to add.

Character to add. In this case, a space.

String Functions

Accessing Individual Characters

Chars is an array of characters that holds all the characters in the string. You can use this property to read individual characters from a string based on their location in the string (the index of the first character in the array is zero). The Chars array is read-only, and you can't edit a string by setting individual characters.

```
Dim myString As String
Dim myChar As Char
```

To get the first character of myString use ...

```
myChar = myString.Chars(0)
```

To get the second character of myString use ...

```
myChar = myString.Chars(1)
```

To get the third character of myString use ...

```
myChar = myString.Chars(2)
```

To get the last character of myString use ...

```
myChar = myString.Chars
                (myString.Length() - 1)
```

Note: Put this all on one line.

Char Methods

Several methods exist that allow the programmer to test certain properties of a character.

```
Dim myChar As Char
```

You can check if the character is a letter, a digit, uppercase, lowercase, etc.

```
Char.IsLetter(myChar)

Char.IsDigit(myChar)

Char.IsLetterOrDigit(myChar)

Char.IsLower(myChar)

Char.IsUpper(myChar)

Char.IsNumber(myChar)
```

Substrings

Substring returns a new string that is a substring of the original string.

The substring begins at the specified index and extends up to the specified length.

```
myString.Substring(___, ___)
```

Starting location

How many characters to extract.

Note: Recall that the first character in the string is in position "0", not "1".

If you want to take the first 5 characters of a string ...

```
myString.Substring(0, 5)
```

If you want to take the third to seventh characters of a string, use ...

```
myString.Substring(2, 5)
```

If you want to take the last character of a string ...

```
myString.Substring(myString.Length - 1, 1)
```

If you want to take the last 5 characters of a string ...

```
myString.Substring(myString.Length - 5, 5)
```

String Functions Examples

Searching for Characters

The following example uses a **For-Next** loop to search a string for the letter "a".

```
Dim myString As String
Dim found As Boolean

myString = "Programming is Fun"
found = False

For i = 0 to myString.Length() - 1

      If myString.Chars(i) = "a" then found = True

Next i
```

We can modify the program to check for uppercase A's as well ...

```
      If myString.Chars(i) = "a" or myString.Chars(i) = "A" then found = True
```

Validating a Password

The following function allows the user to create a password that follows the following rules:

- Must be at least 6 characters
- Must be only numbers and letters

The program assumes that the user has typed the password in TextBox1.

```
Dim password As String
Dim i As Integer
Dim valid As Boolean

password = TextBox1.Text
valid = True    'assume the password is valid

If password.Length < 6 Then
    MessageBox.Show("The password must be at least 6 characters long.")
    valid = false
    Exit Sub
End If

For i = 0 To password.Length - 1

    If Not Char.IsLetterOrDigit(password.Substring(i, 1)) Then valid = False

Next

If valid = False Then MessageBox.Show("Password must be only letters or digits.")
```

TextBox Control - Sample Code

Toggle Bold Text Using a CheckBox

You can use a CheckBox (with the **Appearance** property set to **Button**) to turn Bold on and off (toggle) in a TextBox.

```
If CheckBox1.Checked = True Then
    TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style + FontStyle.Bold)
Else
    TextBox1.Font = New Font(TextBox1.Font, TextBox1.Font.Style - FontStyle.Bold)
End If
```

You can do the same with other font properties such as: FontStyle.Italic, FontStyle.Strikeout, FontStyle.Underline

Adjusting Font Size with a ComboBox

Assuming we have a ComboBox Control populated with numbers (ie. 8, 10, 12, 14, 18, 24, 36, 72).

In Form | Load, type:

```
ComboBox1.Items.Add(8)
ComboBox1.Items.Add(10)
ComboBox1.Items.Add(12)
```
etc.

In the ComboBox1 | SelectedIndexChanged, type:

```
TextBox1.Font = New Font(TextBox1.Font.FontFamily, ComboBox1.SelectedItem)
```

Setting up a "Font" ComboBox

This will get all of the available fonts on the computer and add them to the ComboBox2.

In Form | Load, type:

```
Dim f As System.Drawing.Text.InstalledFontCollection = New System.Drawing.Text.InstalledFontCollection

For Each FontFamily In f.Families
   ComboBox2.Items.Add(FontFamily.Name)
Next
```

In the ComboBox2 | SelectedIndexChanged, type:

```
TextBox1.Font() = New Font(ComboBox2.SelectedItem.ToString, ComboBox1.SelectedItem)
```

Using the FontDialog Control

You can use the FontDialog Control to set many font properties. Begin by adding the FontDialog Control to your Form.

```
FontDialog1.ShowDialog()
TextBox1.Font = FontDialog1.Font
```

Undo

```
If TextBox1.CanUndo Then
   TextBox1.Undo()
End If
```

Select All

```
TextBox1.SelectAll()
```

Using the Clipboard with Text

The Clipboard - Cut

```
My.Computer.Clipboard.Clear()

If TextBox1.SelectionLength > 0 Then TextBox1.Cut()
```

The Clipboard - Copy

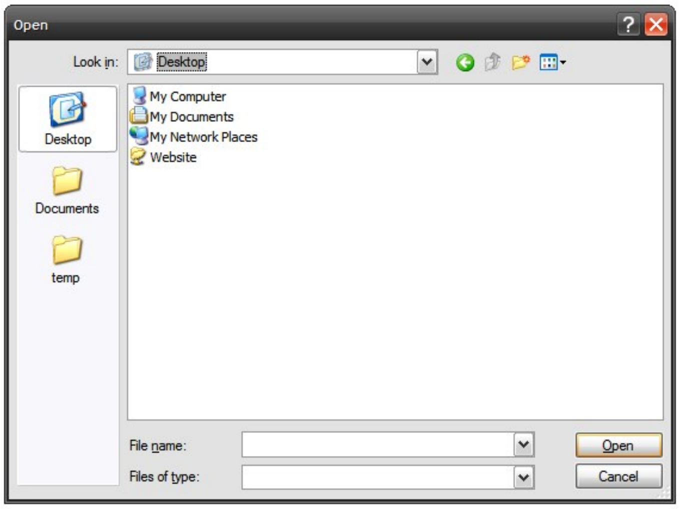
```
My.Computer.Clipboard.Clear()

TextBox1.Copy()
```

The Clipboard - Paste

```
If My.Computer.Clipboard.ContainsText Then TextBox1.Paste()
```

OpenFileDialog Control



Visual Basic provides several dialog controls to facilitate coding and keep your programs consistent with other Windows applications.

The OpenFileDialog box displays a dialog that allows a user to change drives and folders and select a file.

Note: When you add the OpenFileDialog control to your Form, it will appear in a special area below the Form (it will not appear on the actual Form).

To display the OpenFileDialog control ...

```
OpenFileDialog.Show()
```

To determine the file the user selected ...

```
OpenFileDialog.FileName
```

To determine what type of files you want to display, use the **Filter** property ...

Example. Show only text (.TXT) files.

```
OpenFileDialog.Filter = "Text Files (*.txt)|*.txt"
```

The blue text is what the user will see.

The orange text is what the computer will search for.

This character is usually above the [ENTER] key on the keyboard.

Example. Show web page files (can be .HTML or .HTM)

```
SaveFileDialog1.Filter = "HTML Files (*.htm, *.html)|*.htm;*.html"
```

Example. Show text files and web page files.

```
SaveFileDialog1.Filter = "HTML Files (*.htm, *.html)|*.htm;*.html|
            Text Files (*.txt)|*.txt"
```

Note: Put all this on one line.

SaveFileDialog Control

The **SaveFileDialog** control works almost identically to the OpenFileDialog control.

Writing Text to a File

The following code is from "**Mr. G's HTML Editor 2009**".

The name of the document is stored in a string variable named **HTMLFileName**.

A boolean variable named **dirty** determines if the text has been modified since the last time the document was saved. This variable is set to **False** whenever a new document is started, a document is opened, or immediately after a document is saved. In `TextBox1.TextChanged`, the variable is set to **True**.

In the "Save" Command button ...

```
If HTMLFileName = "untitled.html" Then

    SaveFileDialog1.Filter = "HTML Files (*.htm, *.html)|*.htm;*.html|
            Text Files (*.txt)|*.txt"

    SaveFileDialog1.FileName = HTMLFileName
    SaveFileDialog1.ShowDialog()

    HTMLFileName = SaveFileDialog1.FileName
    Me.Text = "Mr. G's HTML Editor - " + HTMLFileName

End If

My.Computer.FileSystem.WriteAllText(HTMLFileName, TextBox1.Text, False)

TextBox1.Focus()

dirty = False
```

Note: Put all this on one line.

In TextBox1 | TextChanged ...

```
dirty = True
```

Reading Text from a File

The following code is from "**Mr. G's HTML Editor 2009**".

The name of the document is stored in a string variable named **HTMLFileName**.

A boolean variable named **dirty** determines if the text has been modified since the last time the document was saved. This variable is set to **False** whenever a new document is started, a document is opened, or immediately after a document is saved. In `TextBox1.TextChanged`, the variable is set to **True**.

In the "Open" Command button ...

```
'Check if the text in the TextBox has been modified

If dirty = True Then

    'If the text of notepad changed, the program will ask the
    'user if they want to save the changes

    response = MessageBox.Show("Do you want to save the changes?",
            "Open Document", MessageBoxButtons.YesNoCancel)
```

Note: Put all this on one line.

```
    If response = DialogResult.Cancel Then Exit Sub

    If response = DialogResult.Yes Then

        SaveFileDialog1.Filter = "HTML Files (*.htm, *.html)|*.htm,*.html|
                  Text Files (*.txt)|*.txt"
```

Note: Put all this on one line.

```
        SaveFileDialog1.FileName = HTMLFileName
        SaveFileDialog1.ShowDialog()

        My.Computer.FileSystem.WriteAllText(SaveFileDialog1.FileName,
                  TextBox1.Text, False)
```

Note: Put all this on one line.

```
        TextBox1.Clear()

    End If

End If

OpenFileDialog1.Filter = "HTML Files (*.htm, *.html)|*.htm;*.html|
                  Text Files (*.txt)|*.txt"
```

Note: Put all this on one line.

```
OpenFileDialog1.FileName = ""
OpenFileDialog1.ShowDialog()

If OpenFileDialog1.FileName = "" Then Exit Sub

TextBox1.Text = My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
TextBox1.Focus()

HTMLFileName = OpenFileDialog1.FileName
Me.Text = "Mr. G's HTML Editor - " + HTMLFileName

dirty = False
```

Text and Strings Exercises

______________________________	______________________________
Name	Date

Given the string … `myString = "Hello, how are you ?"` … what do the following return …

myString.subString(0, 1)	______________________
myString.subString(0, 3)	______________________
myString.subString(7, 3)	______________________
myString.subString(myString.Length() - 1, 1)	______________________
myString.subString(myString.Length() - 5, 5)	______________________

What **substring( )** statement would yield the following results …

"how"	myString.subString(_____, _____)
"are"	myString.subString(_____, _____)
"yo"	myString.subString(_____, _____)

State the command that …

… returns the **length** (number of characters) of a string	______________________
… converts all characters in a string to uppercase	______________________
… converts all characters in a string to lowercase	______________________
… removes leading spaces from the start of a string	______________________
… removes trailing spaces from the end of a string	______________________
… removes leading and trailing spaces from a string	______________________