**Substring method:**

The substring function is used to obtain a part of a specified string. This method is defined in the String class of Microsoft VB.NET. We have to specify the start index from which the String will be extracted. The String will be extracted from that index up to the length that we specify.

**Syntax of Substring**

The function accepts two arguments as shown in the following syntax:

Public Function Substring(ByVal start\_Index As Integer, ByVal sub\_length As Integer) As String

**Here,**

* The ByVal keyword denotes pass-by-value, which is a mechanism of passing arguments to functions.
* The start\_Index is the index from which the substring will be obtained
* sub\_length denotes the length up to which the String will be copied from the start\_Index. This length is measured in terms of the number of characters. The function will return the extracted substring.

**Example:**

Module Module1

Sub Main()

Dim st As String = "central unversity"

Dim subst As String = st.Substring(0, 4)

Console.WriteLine("The substring is: {0}", subst)

Console.ReadKey()

End Sub

End Module

Output:

The substring is: cent

**Trim method:**

The string **Trim** method is used to remove all leading and trailing whitespace characters from the specified string object in visual basic.

 Using the **Trim()** method, we can also remove all occurrences of specified characters from the start and end of the current string.

**String Trim Method Syntax**

Following is the syntax of defining a string **Trim** method to remove all leading and trailing whitespaces or specified characters from the string object in a visual basic programming language.

Public Function Trim() As String

 Public Function Trim(ByVal trimChars As Char()) As String

The first syntax is used to remove all whitespaces from the specified string's starting and end, and it won’t take any parameters.

 The second syntax is used to remove trailing and leading occurrences of the specified characters in an array from the current string object.

**String Trim() Method Example**

Following is the example of using a string **Trim()** method to remove the starting and end of whitespaces or specified characters from the string object in a visual basic programming language.

Module Module1

    Sub Main()

        Dim str1 As String = "  Welcome"

        Dim str2 As String = "    to    "

        Dim str3 As String = "    Central University"

        Console.WriteLine("Before Trim: {0} {1} {2}", str1, str2, str3)

        Console.WriteLine("After Trim: {0} {1} {2}", str1.Trim(), str2.Trim(), str3.Trim())

        Dim trimChars As Char() = {"\*"c, "@"c, " "c}

        Dim str4 As String = "@@\*\*  Computer Science  \*\*@"

        Console.WriteLine("Before Trim: {0}", str4)

        Console.WriteLine("After Trim: {0}", str4.Trim(trimChars))

        Console.WriteLine("Press Enter Key to Exit..")

        Console.ReadLine()

    End Sub

End Module

**String Equals Method**

In visual basic, the string **Equals** method is useful to check whether the specified two string objects are having the same value or not. If both string object values are equal, then **Equals()** method will return **True** otherwise **False**.

Even, if both string objects are having **Null** value, string **Equals()** method will return **True**.

String Equals Method Syntax

Following is the syntax of defining a string **Equals** method to check whether the given string objects are equal or not in a visual basic programming language.

Public Function Equals(ByVal a As String, ByVal b As String) As Boolean

If you observe the syntax, we will use both defined strings to check whether both the strings are equal or not using the **Equals** method.

String Equals Method Example

Following is the example of using the string **Equals()** method to check whether the given strings are equal or not in a visual basic programming language.

 Module Module1

    Sub Main()

        Dim fname As String = "Computer"

        Dim lname As String = "Science"

        Console.WriteLine("{0} Equals to {1}? : {2}", fname, lname, fname.Equals(lname))

        Dim l\_name As String = "computer"

        Console.WriteLine("{0} Equals to {1}? : {2}", fname, l\_name, fname.Equals(l\_name))

        Dim u\_name As String = "Computer"

        Console.WriteLine("{0} Equals to {1}? : {2}", fname, u\_name, fname.Equals(u\_name))

        Console.WriteLine("Press Enter Key to Exit..")

        Console.ReadLine()

    End Sub

End Module

**String Equals Ignore Case**

Generally, in visual basic the string **Equals()** method will perform case-sensitive string comparison. In case, if we want to perform case insensitive string comparison, then we need to use **OrdinalIgnoreCase** property along with the **Equals** method.

 Following is the example of ignoring the case while comparing the strings in a visual basic programming language.

 Module Module1

    Sub Main()

        Dim fname As String = "Sunabeda"

        Dim l\_name As String = "sunabeda"

        Console.WriteLine("{0} Equals to {1}? : {2}", fname, l\_name, fname.Equals(l\_name, StringComparison.OrdinalIgnoreCase))

        Dim u\_name As String = "Sunabeda"

        Console.WriteLine("{0} Equals to {1}? : {2}", fname, u\_name, fname.Equals(u\_name))

                Console.ReadLine()

    End Sub

End Module

we used “**OrdinalIgnoreCase**” property with the **Equals()** method to perform case insensitive comparison by ignoring the case of characters in a visual basic programming language.

**Replace method**

Replace looks for a substring within a list and replaces the substring with a second substring. When we use Strings in Data there are some situations when we need to replace some character or a string from a sentence.

**Syntax:**

The syntax for Replace String function:

Replace ( Expression as String, Old String, New String)  
Now the expression is the main string from where we want to remove the old string and new string which will replace the old string.

**Example1:**

Sub Module1()

Dim Str As String

Str = Replace("My name is Laxmi", "Laxmi", "Devi")

MsgBox (Str)

End Sub

**Example 2**

Module Program

Sub Main(args As String())

Dim x As String = "aaaaa"

Dim nx As String = x.Replace("a", "b").Replace("b", "c")

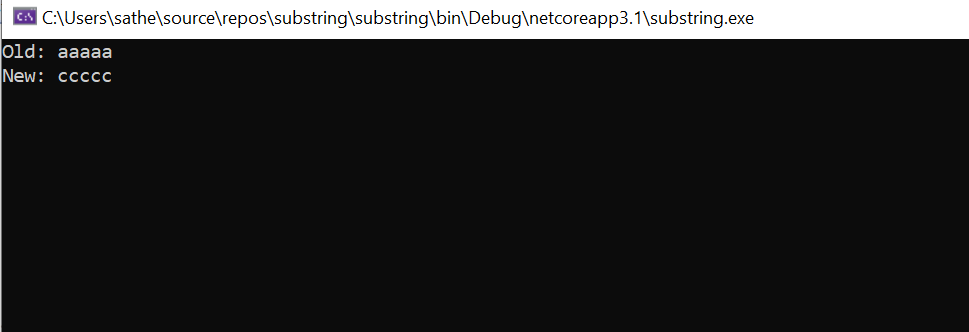
Console.WriteLine("Old: {0}", x)

Console.WriteLine("New: {0}", nx)

Console.ReadLine()

End Sub

End Module



Example 3:

Sub Example2()

Dim Str, Str1, Str2 As String

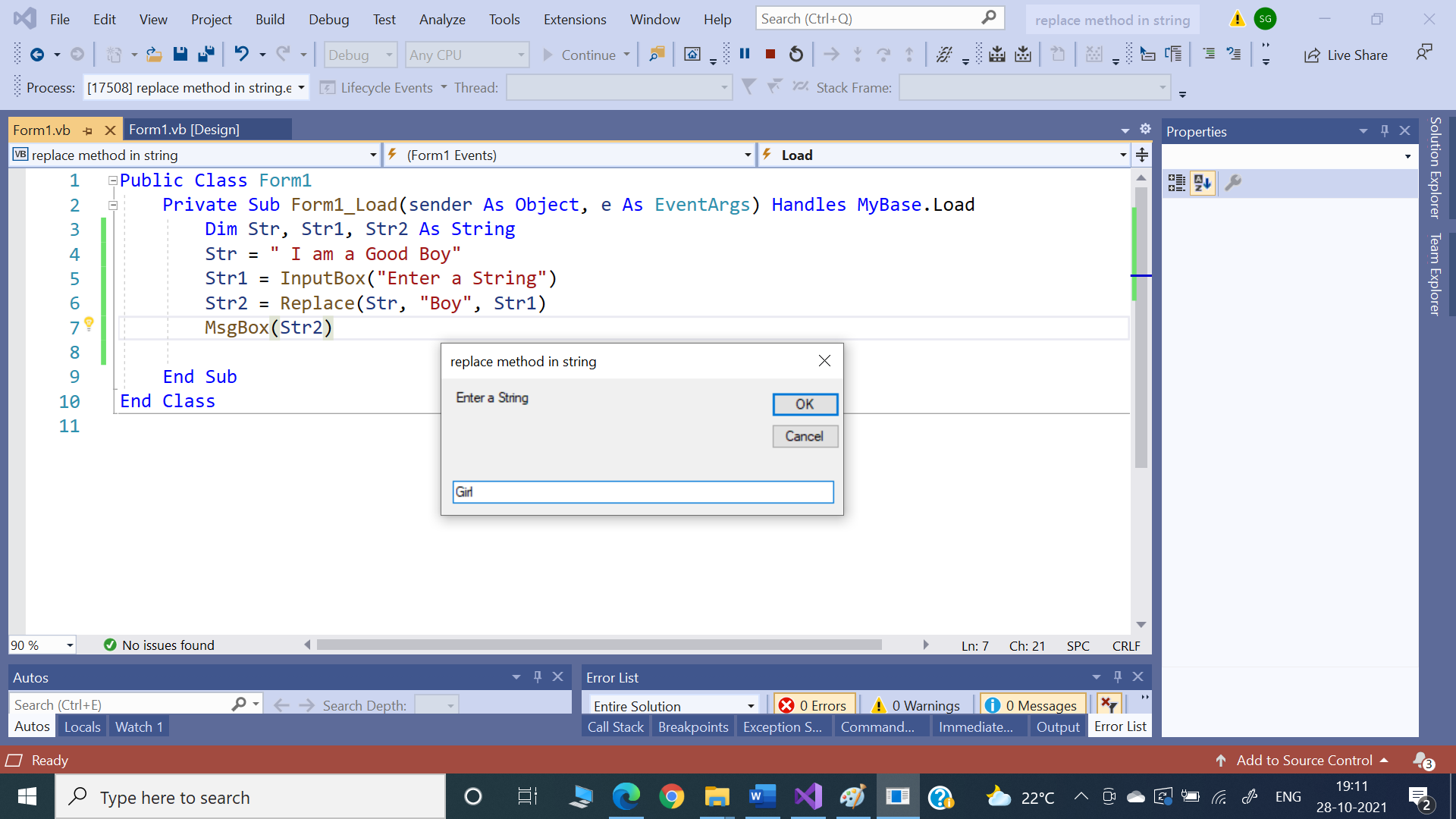
Str = " I am a Good Boy"

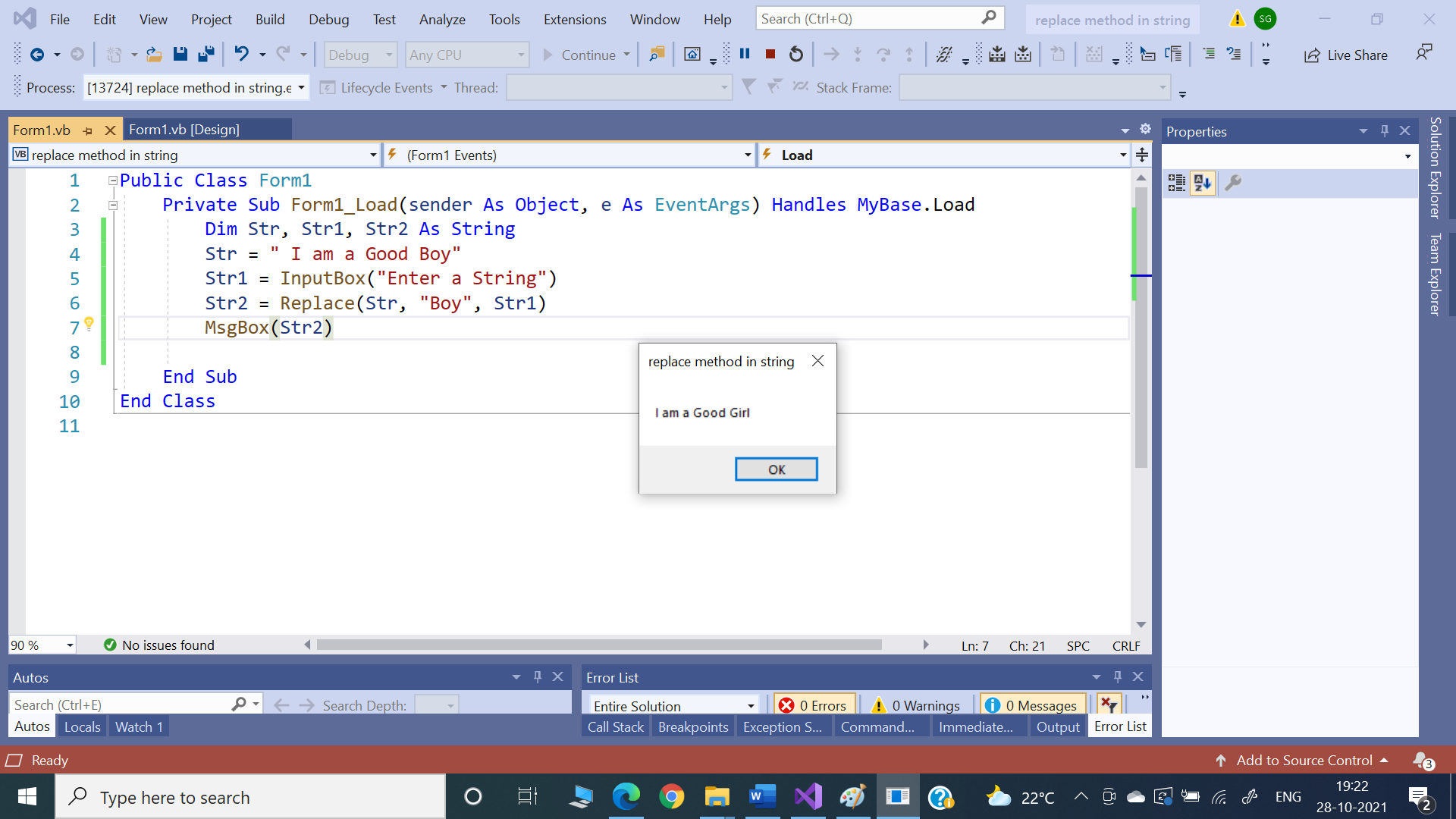
Str1 = InputBox("Enter a String")

Str2 = Replace(Str, "Boy", Str1)

MsgBox (Str2)

End Sub





**String Insert() method**

The **Insert()** method in **String Class** will insert a String in a specified index in the String instance.

String. Insert(Integer ind, String str) as String

* ind - The index of the specified string to be inserted.
* str - The string to be inserted.

Example:

"This is Test".Insert(8,"Insert ") returns "This is Insert Test"

Example 1:

Public Class Form1

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim str As String = "This is VB.NET Test"

Dim insStr As String = "Insert "

Dim strRes As String = str.Insert(15, insStr)

MsgBox(strRes)

End Sub

End Class

**String Split() method**

Split() method extracts the substrings from the given string that are delimited by the separator parameter, and returns those substrings as elements of an array.

If your String contains "dd-mm-yy", split on the "-" character to get an array of: "dd" "mm" "yy".

If the separator parameter is null or contains no characters, white space characters are assumed to be the delimiters.

Public Function Split(ByVal ParamArray separator() As Char) As String()

* separator - the given delimiter.

**Returns:**

* An array of Strings delimited by one or more characters in separator.

**Example 1**

Public Class Form1

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Dim str As String

Dim strArr() As String

Dim count As Integer

str = "vb.net split test"

strArr = str.Split(" ")

For count = 0 To strArr.Length - 1

MsgBox(strArr(count))

Next

End Sub

End Class

Example 2:

Dim input As String = "one)(two)(three)(four)(five"

Dim result As String() = input.Split(New String() {")("}, StringSplitOptions.None)

For Each s As String In result

MessageBox.Show(s)

Next

**Output**

one

two

three

four

five