Data types refer to an extensive system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

### Data Types Available in VB.Net

VB.Net provides a wide range of data types. The following table shows all the data types available:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Storage Allocation</th>
<th>Value Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Depends on implementing platform</td>
<td>True or False</td>
</tr>
<tr>
<td>Byte</td>
<td>1 byte</td>
<td>0 through 255 unsigned</td>
</tr>
<tr>
<td>Char</td>
<td>2 bytes</td>
<td>0 through 65535 unsigned</td>
</tr>
<tr>
<td>Date</td>
<td>8 bytes</td>
<td>0:00:00 midnight on January 1, 0001 through 11:59:59 PM on December 31, 9999</td>
</tr>
<tr>
<td>Decimal</td>
<td>16 bytes</td>
<td>0 through +/-79,228,162,514,264,337,593,543,950,335, for negative values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.94065645841246544E+324, for positive values</td>
</tr>
<tr>
<td>Double</td>
<td>8 bytes</td>
<td>-1.79769313486231570E+308 through +4.94065645841246544E-324, for negative values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.94065645841246544E-324, for positive values</td>
</tr>
<tr>
<td>Integer</td>
<td>4 bytes</td>
<td>-2,147,483,648 through 2,147,483,647 signed</td>
</tr>
<tr>
<td>Object</td>
<td>4 bytes on 32-bit platform</td>
<td>Any type can be stored in a variable of type Object</td>
</tr>
<tr>
<td></td>
<td>8 bytes on 64-bit platform</td>
<td></td>
</tr>
<tr>
<td>SByte</td>
<td>1 byte</td>
<td>-128 through 127 signed</td>
</tr>
<tr>
<td>Short</td>
<td>2 bytes</td>
<td>-32,768 through 32,767 signed</td>
</tr>
<tr>
<td>Single</td>
<td>4 bytes</td>
<td>-3.4028235E+38 through -1.401298E-45 for negative values;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.401298E-45 through 3.4028235E+38 for positive values</td>
</tr>
<tr>
<td>String</td>
<td>Depends on implementing platform</td>
<td>0 to approximately 2 billion Unicode characters</td>
</tr>
</tbody>
</table>
UInteger 4 bytes 0 through 4,294,967,295 unsigned
ULong 8 bytes 0 through 18,446,744,073,709,551,615 unsigned
User-Defined Depends on implementing platform Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort 2 bytes 0 through 65,535 unsigned

Example
The following example demonstrates use of some of the types:

Module DataTypes
    Sub Main()
        Dim b As Byte
        Dim n As Integer
        Dim si As Single
        Dim d As Double
        Dim da As Date
        Dim c As Char
        Dim s As String
        Dim bl As Boolean
        b = 1
        n = 1234567
        si = 0.12345678901234566
        d = 0.12345678901234566
        da = Today
        c = "U"c
        s = "Me"
        If ScriptEngine = "VB" Then
            bl = True
        Else
            bl = False
        End If
        If bl Then
            'the oath taking
            Console.Write(c & " and," & s & vbCrLf)
            Console.WriteLine("declaring on the day of: {0}", da)
            Console.WriteLine("We will learn VB.Net seriously")
            Console.WriteLine("Lets see what happens to the floating point variables:")
            Console.WriteLine("The Single: {0}, The Double: {1}", si, d)
        End If
        Console.ReadKey()
    End Sub
End Module

When the above code is compiled and executed, it produces the following result:

U and, Me
declaring on the day of: 12/4/2012 12:00:00 PM
We will learn VB.Net seriously
Lets see what happens to the floating point variables:
The Single:0.1234568, The Double: 0.123456789012346

The Type Conversion Functions in VB.Net

VB.Net provides the following in-line type conversion functions:

<table>
<thead>
<tr>
<th>S.N</th>
<th>Functions &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
**CBool**\textit{expression}  
Converts the expression to Boolean data type.

**CByte**\textit{expression}  
Converts the expression to Byte data type.

**CChar**\textit{expression}  
Converts the expression to Char data type.

**CDate**\textit{expression}  
Converts the expression to Date data type

**CDBl**\textit{expression}  
Converts the expression to Double data type.

**CDec**\textit{expression}  
Converts the expression to Decimal data type.

**CInt**\textit{expression}  
Converts the expression to Integer data type.

**CLng**\textit{expression}  
Converts the expression to Long data type.

**CObj**\textit{expression}  
Converts the expression to Object type.

**CSByte**\textit{expression}  
Converts the expression to SByte data type.

**CShort**\textit{expression}  
Converts the expression to Short data type.

**CSng**\textit{expression}  
Converts the expression to Single data type.
**CStr**\textit{expression}

Converts the expression to String data type.

**CUInt**\textit{expression}

Converts the expression to UInt data type.

**CULng**\textit{expression}

Converts the expression to ULng data type.

**CUShort**\textit{expression}

Converts the expression to UShort data type.

**Example:**

The following example demonstrates some of these functions:

```
Module DataTypes
    Sub Main()
        Dim n As Integer
        Dim da As Date
        Dim bl As Boolean = True
        n = 1234567
        da = Today
        Console.WriteLine(bl)
        Console.WriteLine(CSByte(bl))
        Console.WriteLine(CStr(bl))
        Console.WriteLine(CStr(da))
        Console.WriteLine(CChar(CChar(CStr(n))))
        Console.WriteLine(CChar(CStr(da)))
        Console.ReadKey()
    End Sub
End Module
```

When the above code is compiled and executed, it produces the following result: