



Working With Numbers

```

Dim i As Integer = 0

i += 10 ' add 10 to i
i -= 5 ' subtract 5 from i
i *= 2 ' multiply i by 2
i /= 2 ' divide i by 2

' parse strings as numbers
' all number types have Parse and TryParse shared functions
Try
    i = Integer.Parse(Console.ReadLine())
    Console.WriteLine("You typed a number")
Catch ex As FormatException
    Console.WriteLine("Please type a number")
End Try

If Integer.TryParse(Console.ReadLine(), i) Then
    Console.WriteLine("You typed a number")
Else
    Console.WriteLine("Please type a number")
End If

' money
Dim price As Decimal = 19.99
price = Decimal.Parse("$39.95" NumberStyles.Currency)

```

Making Decisions

```

Dim count As Integer = 10

' simple 1 line if then else
Dim oneHand As Boolean = IIf(count <= 5, True, False)
If count <= 5 Then oneHand = True Else oneHand = False

' multiple conditions
If count = 0 Then
    Console.WriteLine("Zero")
ElseIf count > 0 And count < 6 Then
    Console.WriteLine("Count on one hand.")
ElseIf count >= 6 And count <= 10 Then
    Console.WriteLine("Count on two hands.")
Else
    Console.WriteLine("Not enough fingers.")
End If

' select case
Select Case count
    Case 0 ' simple value expression
        Console.WriteLine("Zero.")
    Case 1 To 5 ' a range of values
        Console.WriteLine("Count on one hand.")
    Case 6, 7 To 9, 10 ' multiple expressions
        Console.WriteLine("Count on two hands.")
    Case Is > 10, Is <= 20 ' use comparison operators
        Console.WriteLine("Count on hands and feet.")
    Case Else
        Console.WriteLine("Not enough fingers and toes.")
End Select

```

Working With Strings

```

Dim s As String = "Hello World"
Debug.Assert(s.Contains("World"))
Debug.Assert(7 = s.IndexOf("World"))
Debug.Assert(s.StartsWith("Hello"))
Debug.Assert(s.Substring(0, 5) = "Hello")
' combining and formatting strings
s = "Hello" + " " + "World"
s = String.Format("s is: {0}", s)
Dim sb As New StringBuilder()
For i As Integer = 0 To 100
    sb.AppendFormat("{0}", i)
Next
s = sb.ToString()

```

Working With Dates and Times

```

Dim now As Date = Date.Now ' date and time
Dim today As Date = Date.Today ' just the date
Dim tomorrow As Date = today.AddDays(1)
Dim yesterday As Date = today.AddDays(-1)

' use parse functions to convert from strings
Dim vb1 As Date = Date.Parse("1991-05-01")

' use culture info to parse dates in regional formats
Dim gb As New CultureInfo("en-GB")
vb1 = Date.Parse("01-05-1991", gb.DateTimeFormat)

' format dates as strings
vb1.ToLongDateString() ' Wednesday, May 01, 1991
vb1.ToShortDateString() ' 5/1/1991
vb1.ToLongTimeString() ' 12:00:00 AM
vb1.ToShortTimeString() ' 12:00 AM
vb1.ToString("u") ' 1991-05-01 00:00:00Z
vb1.ToString("yyyy-MM-dd") ' 1991-05-01

' TimeSpan represents difference between two dates
Dim time As TimeSpan = tomorrow - today
Dim days As Integer = time.Days
Dim hours As Integer = time.Hours
Dim minutes As Integer = time.Minutes
Dim seconds As Integer = time.Seconds
Dim milliseconds As Integer = time.Milliseconds

' Add time using other timespan instances
time.Add(New TimeSpan(days, hours, minutes, seconds, milliseconds))
' or
time += New TimeSpan(days, hours, minutes, seconds, milliseconds)

' use a timespan to add time to a date
today.Add(time)
' or
today += time
' timespan represented as Days:Hours:Minutes:Seconds:
' Subseconds
time = TimeSpan.Parse("1.2:3:4:5")
' or
If TimeSpan.TryParse(Console.ReadLine(), time) Then
    Console.WriteLine(time)
End If

```

Making Comparisons

```

Debug.Assert(1 = 1) ' equal
Debug.Assert(1 <> 2) ' not equal
Debug.Assert(1 <= 2) ' less than or equal
Debug.Assert(1 >= 0) ' greater than or equal
Debug.Assert(1 < 2) ' less than
Debug.Assert(1 > 0) ' greater than

```

Working With Objects

```

Dim o As Object = "Hello World"
' check type of object
Debug.Assert(InstanceOf o Is String)
' check if reference is not set
Debug.Assert(Not o Is Nothing)

```

Loops

```

For i As Integer = 10 To 1 Step -1
    Console.WriteLine(i)
    If i < 5 Then
        Exit For
    End If
Next
Dim start As Integer = 5
For i As Integer = start To start + 10
    Console.WriteLine(i)
Next
Dim str As String
Do
    str = Console.ReadLine()
    Console.WriteLine(str)
Loop Until str = ""
Do
    str = Console.ReadLine().Trim()
    If String.IsNullOrEmpty(str) Then
        Exit Do
    End If
    Console.WriteLine("You typed:{0}", str)
Loop
Dim names As String() = New String() {"Bob", "David"}
For Each name As String In names
    Console.WriteLine(name)
Next

```



Working With Text

```
Imports System.IO
Imports System.Text

Dim name As String = "David"
Dim hello1 As String = "Hello " + name
Dim hello2 As String = String.Format("Hello {0}", name)
Console.WriteLine("Hello {0}", name)
Dim sb As New StringBuilder("Hello ")
sb.AppendLine(name)
sb.AppendFormat("Goodbye {0}", name)
Console.WriteLine(sb.ToString())

' create a text file
Using w As StreamWriter = File.CreateText("c:\names.txt")
    w.WriteLine("Bob")
    w.WriteLine("David")
End Using

' read from a text file
Using r As StreamReader = File.OpenText("c:\names.txt")
    Do While Not r.EndOfStream
        Console.WriteLine(r.ReadLine())
    Loop
End Using
For Each s As String In File.ReadAllLines("c:\names.txt")
    Console.WriteLine(s)
Next
```

Working With XML

```
Imports System.IO
Imports System.Xml

' create an xml file
Using xw As XmlWriter = XmlWriter.Create("c:\names.xml")
    xw.WriteRaw("<names>")
    xw.WriteRaw("<name>Bob</name>")
    xw.WriteRaw("<name>David</name>")
    xw.WriteRaw("</names>")
End Using

' load an xml file
Dim doc As New XmlDocument()
doc.Load("c:\names.xml")
For Each node As XmlNode In doc.SelectNodes("//name/text()")
    Console.WriteLine(node.Value)
Next
```

Working With a Database

```
Imports System.Data
Imports System.Data.SqlClient

Dim cs As String = "Data Source=.\SQLEXPRESS;" + _
    "Initial Catalog=NamesDB;" + _
    "Integrated Security=True;"

Using con As New SqlConnection(cs)
    con.Open()
    ' insert a record
    sql = "INSERT INTO Names(Name) VALUES(@Name)"
    Dim cmd1 As New SqlCommand(sql, con)
    cmd1.Parameters.Add("@Name", SqlDbType.NVarChar, 100)
    cmd1.Parameters("@Name").Value = "Bob"
    cmd1.ExecuteNonQuery()
    ' insert a second record
    cmd1.Parameters("@Name").Value = "David"
    cmd1.ExecuteNonQuery()
    ' read records
    sql = "SELECT * FROM Names"
    Dim cmd2 As New SqlCommand(sql, con)
    Using r As SqlDataReader = cmd2.ExecuteReader()
        Dim iName As Integer = r.GetOrdinal("Name")
        Do While r.Read()
            If r.IsDBNull(iName) Then
                Console.WriteLine("Null")
            Else
                Console.WriteLine(r.GetString(iName))
            End If
        Loop
    End Using
    ' read a single value
    sql = "SELECT TOP 1 Name FROM Names"
    Dim cmd3 As New SqlCommand(sql, con)
    Console.WriteLine(cmd3.ExecuteScalar())
End Using
```

Classes

```
' Fields, properties, methods and constructors
Public Class SomeClass
    ' field
    Private mName As String
    ' read/write property
    Public Property Name() As String
        Get
            Return mName
        End Get
        Set(ByVal value As String)
            mName = value
        End Set
    End Property
    ' read only property
    Public ReadOnly Property FormattedName() As String
        Get
            Return String.Format("Hello {0}", mName)
        End Get
    End Property
    ' methods/procedures
    Public Sub PrintName(ByVal format As String)
        Console.WriteLine(format, Me.FormattedName)
    End Sub
    ' constructor
    Sub New(ByVal name As String)
        mName = name
    End Sub
End Class

' Inheritance/Subclassing
Public Class BaseClass
    Public Overridable Sub SayHello()
        Console.WriteLine("Hello BaseClass")
    End Sub
End Class
Public Class DerivedClass
    Inherits BaseClass
    ' override inherited method
    Public Overrides Sub SayHello()
        ' invoke the base class version of the method
        MyBase.SayHello()
        Console.WriteLine("Hello DerivedClass")
    End Sub
End Class
```