Unit I

- Introduction to .NET, 4.0.NET Framework features & architecture. Introduction to Visual Studio2010, Event Driven Programming, VB.NET Development Environment, Solution Explorer, Toolbox, Properties Window, Form Designer, Output Window, Object Browser. The VB.NET Language - data types, variables, variables declarations, Scope of a variable, type casting, constants, operators and expressions.

Introduction to .NET Framework

- The **.NET Framework** is a software development platform
- Introduced by Microsoft in the late 1990 under the NGWS.
- On 13 February 2002, Microsoft launched the first version of the .NET Framework, referred to as the **.NET Framework 1.0**.

What is .NET Framework

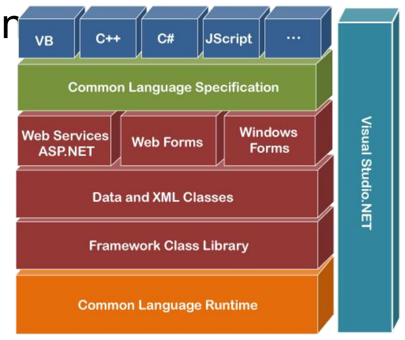
- It is a virtual machine that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc.
- It is also used to create a form based, console-based, mobile and web-based application or services that are available in Microsoft environment.
- Furthermore, the .NET framework is a pure object oriented, that similar to the Java language

What is .NET Framework

- But it is not a platform independent as the Java. So, its application runs only to the windows platform.
- The main objective of this framework is to develop an application that can run on the windows platform.
- The current version of the .Net framework is 4.8.
- The .NET Framework is not only a language, but it is also a software and language neutral platform.

Components of .NET Framework

- CLR (Common Language Runtime)
- CTS (Common Type System)
- BCL (Base Class Library)
- CLS (Common Language Specification)
- FCL (Framework Class Library)
- .NET Assemblies
- XML Web Services
- Window Services



CLR (Common Language Runtime)

- It is an important part of a .NET framework that works like a virtual component of the .NET Framework to executes the different languages program like c#, Visual Basic, etc.
- A CLR also helps to convert a source code into the byte code, and this byte code is known as CIL (Common Intermediate Language) or MSIL (Microsoft Intermediate Language).
- After converting into a byte code, a CLR uses a JIT compiler at run time that helps to convert a CIL or MSIL code into the machine or native code.

CTS (Common Type System)

- It specifies a standard that represent what type of data and value can be defined and managed in computer memory at runtime.
- A CTS ensures that programming data defined in various languages should be interact with each other to share information.
- For example, in C# we define data type as int, while in VB.NET we define integer as a data type.

BCL (Base Class Library)

- The base class library has a rich collection of libraries features and functions that help to implement many programming languages in the .NET Framework, such as C #, F #, Visual C ++, and more. Furthermore, BCL divides into two parts:

User defined class library

- **Assemblies -** It is the collection of small parts of deployment an application's part. It contains either the DLL (Dynamic Link Library) or exe (Executable) file.
  - In DLL, it uses code reusability, whereas in exe it contains only output file/ or application.
  - DLL file can't be open, whereas exe file can be open.
  - DLL file can't be run individually, whereas in exe, it can run individually.
  - In DLL file, there is no main method, whereas exe file has main method.

Predefined class library

- **Namespace -** It is the collection of predefined class and method that present in .Net.
- In other languages such as, C we used header files, in java we used package similarly we used "using system" in .NET, where using is a keyword and system is a namespace.

CLS (Common language Specification)

- It is a subset of common type system (CTS) that defines a set of rules and regulations which should be followed by every language that comes under the .net framework.
- In other words, a CLS language should be cross-language integration or interoperability.
- For example, in C# and VB.NET language, the C# language terminate each statement with semicolon, whereas in VB.NET it is not end with semicolon, and when these statements execute in .NET Framework, it provides a common platform to interact and share information with each other.

Microsoft .NET Assemblies

- A .NET assembly is the main building block of the .NET Framework.
- It is a small unit of code that contains a logical compiled code in the Common Language infrastructure (CLI), which is used for deployment, security and versioning.
- It defines in two parts (process) DLL and library (exe) assemblies.
- When the .NET program is compiled, it generates a metadata with Microsoft Intermediate Language, which is stored in a file called Assembly.

FCL (Framework Class Library)

- It provides the various system functionality in the .NET Framework, that includes classes, interfaces and data types, etc.
- to create multiple functions and different types of application such as desktop, web, mobile application, etc.
- In other words, it can be defined as, it provides a base on which various applications, controls and components are built in .NET Framework.

Key Components of FCL

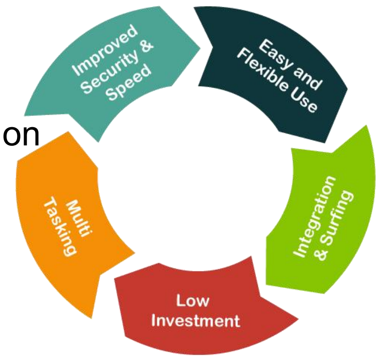
- Object type
- Implementation of data structure
- Base data types
- Garbage collection
- Security and database connectivity
- Creating common platform for window and web-based application

Versions of .NET Framework

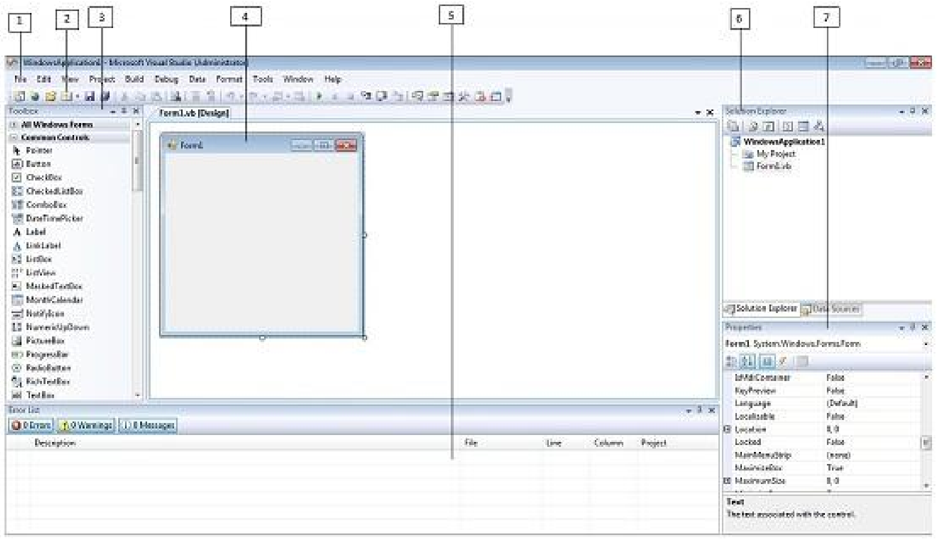
- On 13 February 2002, Microsoft launched first version of .Net framework 1.0.
- The second version 2.0 of .net framework was launched on 22 January 2006.
- Third version 3.0 of .Net framework was released on 21 November 2006.
- A .Net framework version 3.5 was released on 19 November 2007.
- Version 4.0 of .Net framework was released on 29 September 2008
- Version 4.5 of .Net framework was released on 15 August 2012.
- .Net framework 4.5.1 version was announced on 17 October 2013
- On 5 May 2014, a 4.5.2 version of .Net framework was released.
- .Net framework 4.6 version was announced on 12 November 2014
- .Net framework 4.6.1 version was released on 30 October 2015
- .Net framework 4.6.2 version was announced on March 30, 2016
- .Net framework 4.7 version was announced on April 5, 2017
- .Net framework 4.7.1 version was announced on October 17, 2017
- Version 4.7.2 of .Net framework was released on 30 April 2018.
- And currently we are using .Net framework version 4.8 that was released on 18 April 2019

Characteristics of .NET Framework

- CLR (Common Language Runtime)
- Namespace - Predefined class and function
- Metadata and Assemblies
- Application domains
- It helps to configure and deploy the .net application
- It provides form and web-based services
- NET and ASP.NET AJAX
- LINQ
- Security and Portability
- Interoperability
- It provides multiple environments for developing an application



Visual Studio IDE



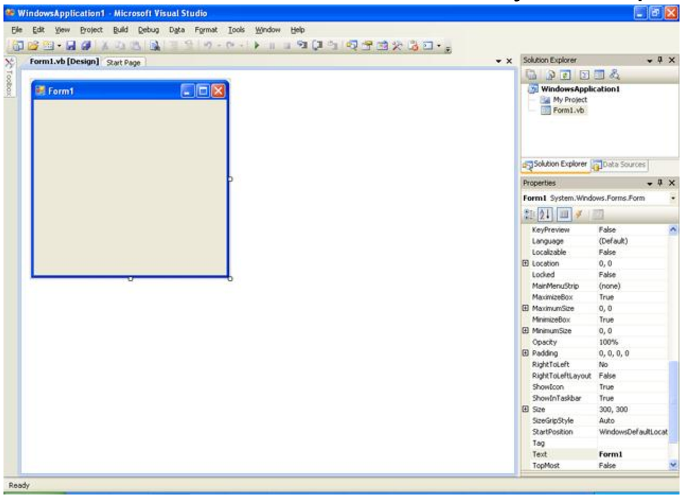
Property	Value
IsMdiContainer	False
KeyPreview	False
Language	(Default)
Localizable	False
Location	0, 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True
MaximumSize	0, 0

Text
The text associated with the control.

1. Menu Bar
2. Standard Toolbar
3. ToolBox
4. Forms Designer
5. Output Window
6. Solution Explorer
7. Properties Window

Visual Basic.net 2010 IDE

- ***Integrated Development Environment (IDE)*** consist of inbuilt compiler, debugger, editors, automation tools for easy development of code.



Form1 System.Windows.Forms.Form

KeyPreview	False
Language	(Default)
Localizable	False
⊞ Location	0, 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True
⊞ MaximumSize	0, 0
MinimizeBox	True
⊞ MinimumSize	0, 0
Opacity	100%
⊞ Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
⊞ Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocat
Tag	
Text	**Form1**
TopMost	False

Menu Bar

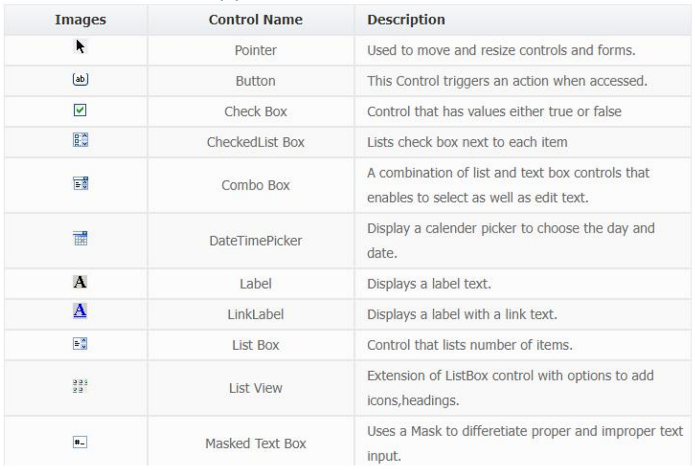
- ***Menu bar*** in Visual Basic.net 2010 consist of the commands that are used for constructing a software code. These commands are listed as menus and sub menus.
- Menu bar also includes a simple Toolbar, that lists the commonly used commands as buttons. This Toolbar can be customized, so that the buttons can be added as required.

Button	Description
	Adds a new Project.
	Open a New Window.
	Open a File.
	Saves the Current Form.
	Saves all files related to a project.
	Cut the selection.
	Copy the selection.
	Paste the selection.
	Find the searched text.
	Comment out selected lines.
	Uncomment the selected lines.
	Undo.
	Redo.

▶	Continue Debugging.
⏸	Break Debugging.
■	Stop Debugging.
	Displays Solution Explorer.
	Displays Properties Window.
	Displays Object Browser.
	Displays ToolBox Window.
	Displays Error List Window.
	Displays Command Window.

Tool Box

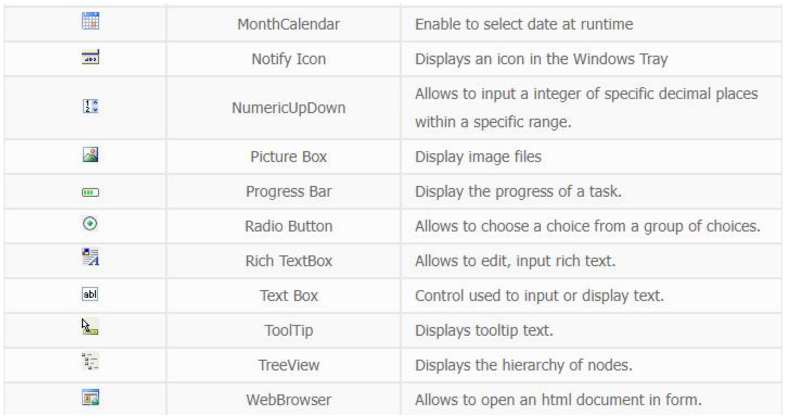
- ***Toolbox*** in Visual Basic.net consist of Controls, Containers, Menu Options, Crystal Report Controls, Data Controls, Dialogs, Components, Printing controls, that are used in a form to design the interfaces of an application.



Images	Control Name	Description
	Pointer	Used to move and resize controls and forms.
	Button	This Control triggers an action when accessed.
	Check Box	Control that has values either true or false
	CheckedList Box	Lists check box next to each item
	Combo Box	A combination of list and text box controls that enables to select as well as edit text.
	DateTimePicker	Display a calender picker to choose the day and date.
	Label	Displays a label text.
	LinkLabel	Displays a label with a link text.
	List Box	Control that lists number of items.
	List View	Extension of ListBox control with options to add icons,headings.
	Masked Text Box	Uses a Mask to differetiate proper and improper text input.

Tool Box

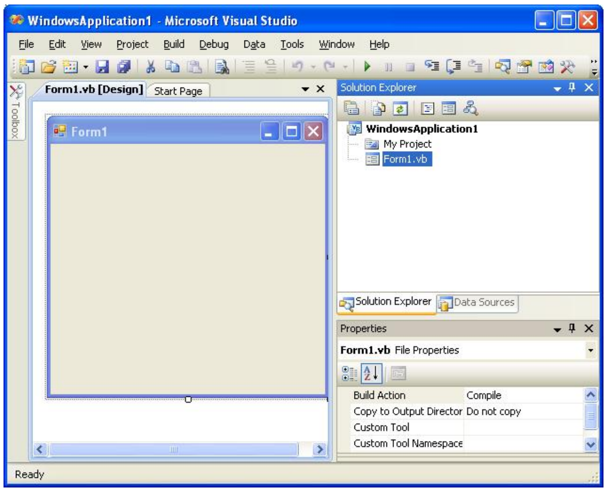
- ***Toolbox*** in Visual Basic.net consist of Controls, Containers, Menu Options, Crystal Report Controls, Data Controls, Dialogs, Components, Printing controls, that are used in a form to design the interfaces of an application.



	MonthCalendar	Enable to select date at runtime
	Notify Icon	Displays an icon in the Windows Tray
	NumericUpDown	Allows to input a integer of specific decimal places within a specific range.
	Picture Box	Display image files
	Progress Bar	Display the progress of a task.
	Radio Button	Allows to choose a choice from a group of choices.
	Rich TextBox	Allows to edit, input rich text.
	Text Box	Control used to input or display text.
	ToolTip	Displays tooltip text.
	TreeView	Displays the hierarchy of nodes.
	WebBrowser	Allows to open an html document in form.

Solution Explorer

- ***Solution Explorer*** in Visual Basic.net lists of all the files associated with a project. It is docked on the right under the Toolbar in the VB IDE. In VB 6 this was known as 'Project Explorer'
- Solution Explorer has options to view the code, form design, refresh listed files. Projects files are displayed in a drop down tree like structure, widely used in Windows based GUI applications.

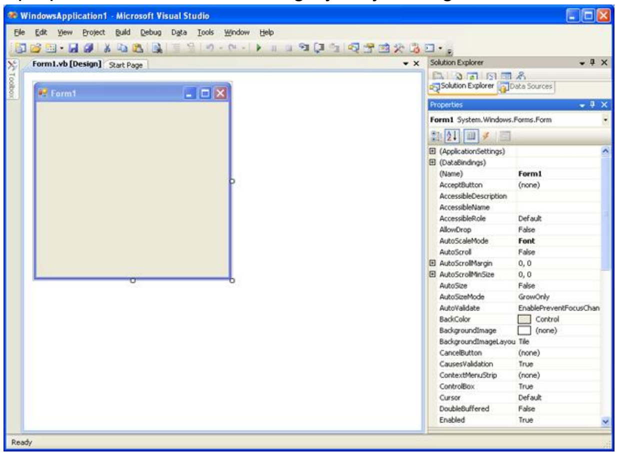


Form1.vb File Properties

Build Action	Compile
Copy to Output Director	Do not copy
Custom Tool	
Custom Tool Namespace	

Properties Window

- ***Windows form properties*** in Visual Basic.net lists the properties of an selected object. Every object in VB has it own properties that can be used to change the look and even the functionality of the object.
- Properties Window lists the properties of the forms and controls in an alphabetical order by default. User also has options to view the properties based on the category they belong to.

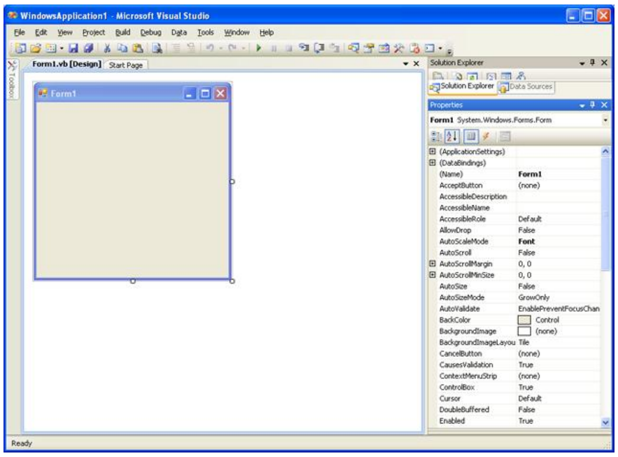

WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Help
Form1.vb [Design]
Start Page
Toolbox
Form1
Solution Explorer
Solution Explorer
Data Sources
Properties
Ready

Form1 System.Windows.Forms.Form

(ApplicationSettings)	
(DataBindings)	
(Name)	Form1
AcceptButton	(none)
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	Font
AutoScroll	False
AutoScrollMargin	0, 0
AutoScrollMinSize	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocusChan
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayou	Tile
CancelButton	(none)
CausesValidation	True
ContextMenuStrip	(none)
ControlBox	True
Cursor	Default
DoubleBuffered	False
Enabled	True

Visual Basic Forms

- Forms in Visual Basic.net are the basic object used to develop an application, it also contains the coding as well as the controls embedded in it.
- **Forms** is created by default when a **Project** is created with a default name **Form1**. Every form has its own Properties, Methods and Events. Usually the form properties name, caption are changed as required, since multiple forms will be used in a Project.

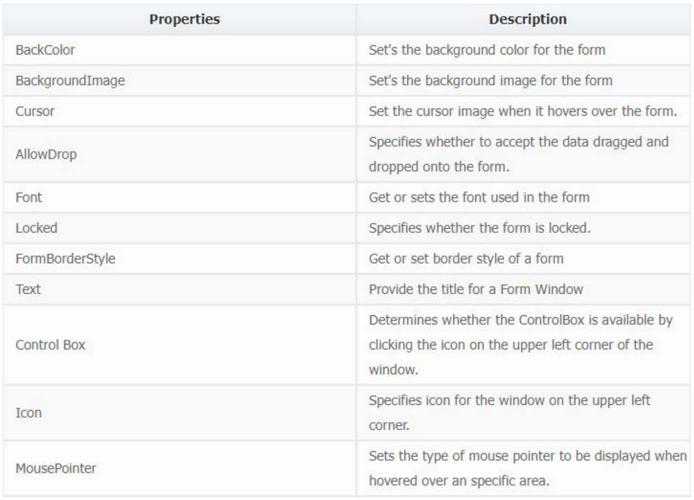

WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Help
Form1.vb [Design]
Start Page
Toolbox
Form1
Solution Explorer
Data Sources
Properties
Ready

Form1 System.Windows.Forms.Form

(ApplicationSettings)	
(DataBindings)	
(Name)	**Form1**
AcceptButton	(none)
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	**Font**
AutoScroll	False
AutoScrollMargin	0, 0
AutoScrollMinSize	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocusChan
BackColor	Control
BackgroundImage	(none)
BackgroundImageLayou	Tile
CancelButton	(none)
CausesValidation	True
ContextMenuStrip	(none)
ControlBox	True
Cursor	Default
DoubleBuffered	False
Enabled	True

Form Properties

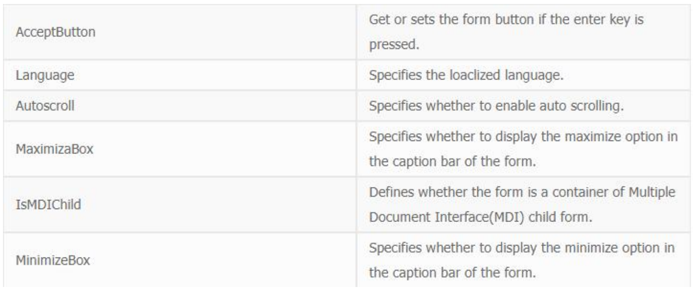
- ***Forms*** in Visual Basic.net is so important to the user as it is the application for the user. So identifying and displaying forms is vital to the usability of an application. Even the developers need to alter the properties to manage the forms in VB.net.



Properties	Description
BackColor	Set's the background color for the form
BackgroundImage	Set's the background image for the form
Cursor	Set the cursor image when it hovers over the form.
AllowDrop	Specifies whether to accept the data dragged and dropped onto the form.
Font	Get or sets the font used in the form
Locked	Specifies whether the form is locked.
FormBorderStyle	Get or set border style of a form
Text	Provide the title for a Form Window
Control Box	Determines whether the ControlBox is available by clicking the icon on the upper left corner of the window.
Icon	Specifies icon for the window on the upper left corner.
MousePointer	Sets the type of mouse pointer to be displayed when hovered over an specific area.

Form Properties

- ***Forms*** in Visual Basic.net is so important to the user as it is the application for the user. So identifying and displaying forms is vital to the usability of an application. Even the developers need to alter the properties to manage the forms in VB.net.



AcceptButton	Get or sets the form button if the enter key is pressed.
Language	Specifies the loaclized language.
Autoscroll	Specifies whether to enable auto scrolling.
MaximizaBox	Specifies whether to display the maximize option in the caption bar of the form.
IsMDIChild	Defines whether the form is a container of Multiple Document Interface(MDI) child form.
MinimizeBox	Specifies whether to display the minimize option in the caption bar of the form.

VB.NET Data Type

- In **VB.NET, data type** is used to define the type of a variable or function in a program. Furthermore, the conversion of one data type to another type using the data conversion function.
- A **Data Type** refers to which type of data or value is assigning to a variable or function so that a variable can hold a defined data type value.
- For example, when we declare a variable, we have to tell the compiler what type of data or value is allocated to different kinds of variables to hold different amounts of space in computer memory.
- **Syntax:** Dim Variable_Name as DataType
- **VariableName:** It defines the name of the variable that you assign to store values.
- **DataType:** It represents the name of the data type that you assign to a variable.

Data Types	**Required Space**	**Value Range**
Boolean	A Boolean type depends on the implementing platform	True or False
Byte	1 byte	Byte Range start from 0 to 255 (unsigned)
Char	2 bytes	Char Range start from 0 to 65535 (unsigned)
Date	8 bytes	Date range can be 0:00:0 (midnight) January 1, 0001 to 11:5959 PM of December 31, 9999.
Decimal	16 bytes	Range from 0 to +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9…E+28) without any decimal point; And 0 to +/-7.9228162514226433759354395033 5 with 28 position to the right of the decimal

VB.NET Data Type

Double	8 bytes	-1.79769313486231570E+308 to -4.94-65645841246544E-324 for negative values; 4.94065645841246544E-324 to 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 to 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 (9.2...E + 18) (signed)
Object	Object size based on the platform such as 4 bytes in 32-bit and 8 bytes in 64-bit platform	It can store any type of data defined in a variable of type Object
SByte	1 byte	-128 to 127 (signed)
Short	2 bytes	-32,768 to 32,767 (signed)
Single	4 bytes	-3.4028235E + 38 to -1.401298E-45 for negative values; And for positive value: 1.401298E-45 to 3.4028235E + 38.
String	String Datatype depend on the implementing platform	It accepts Unicode character from 0 to approximately 2 billion characters.
UInteger	4 bytes	The range start from 0 to 4,294,967,295 (unsigned)
ULong	8 bytes	The range of ULong start from 0 to 18,446,744,073,709,551,615 (1.8...E + 19) (unsigned)
User-Defined (structure)	A user-defined data type depends on the implementing platform	Each member of the structure has its own data type and limits independent of the other members' ranges.
UShort	2 bytes	Range from 0 to 65,535 (unsigned)

Data_type.vb

- Module Data_type
- Sub Main()
- ' defining the Data Type to the variables
- Dim b As Byte = 1
- Dim num As Integer = 5
- Dim si As Single
- Dim db As Double
- Dim get_date As Date
- Dim c As Char
- Dim str As String
- b = 1
- num = 20
- si = 0.12
- db = 2131.787
- get_date = Today
- c = "A"
- str = "Hello Friends..."

Data_type.vb

- Console.WriteLine("Welcome to the BCA")
- Console.WriteLine("Byte is: {0}", b)
- Console.WriteLine("Integer number is: {0}", num)
- Console.WriteLine("Single data type is: {0}", si)
- Console.WriteLine("Double data type is: {0}", db)
- Console.WriteLine("Today is: {0}", get_date)
- Console.WriteLine("Character is: {0}", b)
- Console.WriteLine("String message is: {0}", str)
- Console.ReadKey()
- End Sub
- End Module

What is a Variable?

- A variable is a simple name used to store the value of a specific data type in computer memory.
- In VB.NET, each variable has a particular data type that determines the size, range, and fixed space in computer memory.
- With the help of variable, we can perform several operations and manipulate data values in any programming language.
- VB.NET Variables Declaration
- The declaration of a variable is simple that requires a variable name and data type followed by a Dim.
- A Dim is used in Class, Module, structure, Sub, procedure.

What is a Variable?

- **Syntax:**
- Dim [Variable_Name] As [Defined Data Type]

• **Dim**	It is used to declare and allocate the space for one or more variables in memory.
• **Variable_Name**	It defines the name of the variable to store the values.
• **As**	It is a keyword that allows you to define the data type in the declaration statement.
• **Data Type**	It defines a data type that allows variables to store data types such as Char, String, Integer, Decimal, Long, etc.
• **Value**	Assign a value to the variable.

There are some valid declarations of variables along with their data type definition, as shown below:
Dim Roll_no As Integer
Dim Emp_name As String
Dim Salary As Double
Dim Emp_id, Stud_id As Integer
Dim result_status As Boolean
Further, if we want to **declare more than one variable** in the same line, we must separate each variable with a comma.
Syntax
Dim Variable_name1 As DataType1, variable_name2 As DataType2, Variable_name3 As DataType3
Note: The statements given below is also used to declare the variable with their data type:
Static name As String
Public bill As Decimal = 0

VB.NET Variable Initialization
After the declaration of a variable, we must assign a value to the variable. The following syntax describes the initialization of a variable:
Syntax:
Variable_name = value
For example:
Dim Roll_no As Integer 'declaration of Roll_no
Roll_no = 101 'initialization of Roll_no

Initialize the Emp_name
Dim Emp_name As String
Emp_name = "David" 'Here Emp_name variable assigne
d a value of David

Initialize a Boolean variable
Dim status As Boolean 'Boolean value can be True or Fals
e.
status = True 'Initialize status value to True
We can also initialize a variable at the time of declaration:
Dim Roll_no As Integer = 101
Dim Emp_name As String = " Stephen Robert "

Variable1.vb

```
Imports System
Module Variable1
  Sub Main()
    Dim intData As Integer
    Dim CharData As Char
    Dim strData As String
    Dim dblData As Double
     Dim single_data As Single
     intData = 10
    CharData = "A"
    strData = " VB.NET is a Programming Language."
    dblData = 4567.676
    single_data = 23.08
   Console.WriteLine(" Value of intData is: {0}", intData)
    Console.WriteLine(" Value of CharData is: {0}", CharData)
    Console.WriteLine(" Value of strData is: {0}", strData)
    Console.WriteLine(" Value of dblData is: {0}", dblData)
    Console.WriteLine(" Value of single_data is: {0}", single_data)
    Console.WriteLine("press any key to exit...")
    Console.ReadKey()
  End Sub
 End Module
```

Getting Values from the User in VB.NET
In VB.NET, the Console class provides the Readline() function in the System namespace. It is used to take input from the user and assign a value to a variable. For example:

```
Dim name As String
name = Console.ReadLine()
Or name = Console.ReadLine
```

Let's create a program that takes input from the user.

User_Data.vb

```
Imports System
Module User_Data
  Sub Main()
    Dim num As Integer
    Dim age As Double
    Dim name As String
    Console.WriteLine("Enter your favourite number")
    num = Console.ReadLine
    Console.WriteLine(" Enter Your Good name")
    'Read string data from the user
    name = Console.ReadLine
    Console.WriteLine(" Enter your Age")
    age = Console.ReadLine
    Console.WriteLine(" You have entered {0}", num)
```

VB.NET Constants
As the name suggests, the name constant refers to a fixed value that cannot be changed during the execution of a program. It is also known as **literals**. These constants can be of any data type, such as Integer, Double, String, Decimal, Single, character, enum, etc.
Declaration of Constants
In VB.NET, **const** is a keyword that is used to declare a variable as constant. The Const statement can be used with module, structure, procedure, form, and class.
Syntax:
Const constname As datatype = value

Const	It is a Const keyword to declare a variable as constant.
Constname	It defines the name of the constant variable to store the values.
As	It is a keyword that allows you to define the data type in the declaration statement.
Data Type	It defines a data type that allows variables to store data types such as Char, String, Integer, Decimal, Long, etc.
Value	Assign a value to the variable as constant.

Further, if we want to declare more than one variable in the same line, we must separate each variable with a comma, as shown below. The Syntax for defining the multiple variables as constant is:

Dim Variable_name1 As DataType1, variable_name2 As DataType2, Variable_name3 As DataType3
Const num As Integer = 10
Static name As String
Public Const name As String = "JavaTpoint"
Private Const PI As Double = 3.14

Example of Const keyword
Const1.vb

```
Module Const1
   Sub main()
      Const intData As Integer = 20
      Const name As String = "JavaTpoint"
      Const topic As String = "VB.NET"
      Const PI = 3.14
      Dim radius, area As Integer
      Console.WriteLine(" Constant integer is {0}", intDat
a)
      Console.WriteLine(" You have entered {0}", name)
      Console.WriteLine(" Your Topic is {0}", topic)
      Console.WriteLine("Enter the Radius")
      radius = Console.ReadLine()
      area = PI * radius * radius
      Console.WriteLine(" Area of Circle is {0}", area)
      Console.ReadKey()
   End Sub
End Module
```

Scope of Variable in VB.NET
The scope of a variable determines the accessible range of a defined variable at the time of declaration in any block, module, and class. You can access it if the variable is in a particular region or scope in the same block. And if the variable goes beyond the region, its scope expires.
The following are the methods to represent the scope of a variable in VB.NET.
Procedure Scope
Module Scope
Public Scope

Procedure (local) scope
A **local variable** is a type of variable defined within a procedure scope, block, or function. It is available with a code inside the procedure, and it can be declared using the **Dim or static** statement. These variables are not accessible from outside of the local method. However, the local variable can be easily accessed by the nested programming function in the same method.
Dim X As Integer
Local variables exist until the procedure in which they are declared is executed. Once a procedure is executed, the values of its local variables will be lost, and the resources used by these variables will be released. And when the block is executed again, all the local variables are rearranged.

Local_Scope.vb

```
Imports System
Module Local_scope
  Sub Main()
    Console.WriteLine(" Scope of local varibale within a function")
    local()
    Console.WriteLine("press any key to exit...")
    Console.ReadKey()
  End Sub
  Sub local()
    Dim X As Integer
      X = 50
    Console.WriteLine(" Value of Local value X is {0}", X)
   End Sub
  Sub local2()
    Dim X As String
     X = "JavaTpoint"
    Console.WriteLine(" Value of X is {0}", X)
  End Sub
End Module
```

Module Scope

- All existing procedures can easily identify a variable that is declared inside a module sheet is called a **module-level variable**. The defined module variable is **visible to all procedure**s within that module only, but it is not available for other module's procedures. The **Dim or private statement** at the top of the first procedure declaration can be declared the module-level variables. It means that these variables cannot be declared inside any procedure block. Further, these variables are useful to share information between the procedures in the same module. And one more thing about the module-level variable is that these variables can remains existence as long as the module is executed.
- ' It is the declaration section of the module
- Private num As Integer ' A **private** module-level variable
- Dim name As String ' Another **private** module-level variable