AMIT ACADEMY
for Computer Education

Subject – Desktop Application Development (VB.NET)

Unit-2
Programming Concept

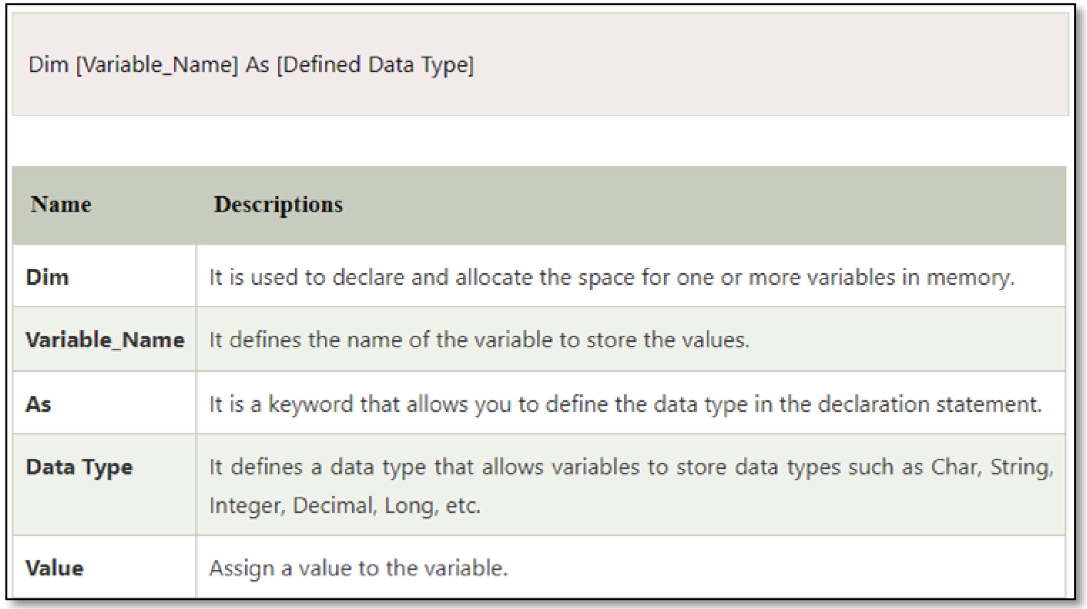
AMIT ACADEMY FOR COMPUTER EDUCATION

Nr Vardayini Mataji Temple, Rupal, Gandhinagar-382630

Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

Variable declaration

- **Variable** is used to hold the value that can be used further in the programming. In this section, we will learn how to declare and initialize a **variable** and a **constant**.
- 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**
  - Dim Roll_no As Integer = 5
  - Dim Emp_name As String
  - Dim Salary As Double
  - Salary = 29000
  - Dim Emp_id, Stud_id As Integer
  - Dim result_status As Boolean



```
Dim [Variable_Name] As [Defined Data Type]
```

Name	Descriptions
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.

Reference Link –

https://www.javatpoint.com/vb-net-variable-and-constant

https://www.tutorialspoint.com/vb.net/vb.net_variables.htm

Operator in VB.Net

- **operator** is a special symbol that tells the compiler to perform the specific logical or mathematical operation on the data values. The data value itself (which can be either a variable or a constant) is called an **operand,** and the Operator performs various **operations** on the operand.
- **Operators in VB.NET**
  - **Arithmetic Operators**
  - **Assignment Operators**
  - **Comparison Operators**
  - **Logical/Bitwise Operators**
  - **Concatenation Operators**
  - **Miscellaneous Operators**

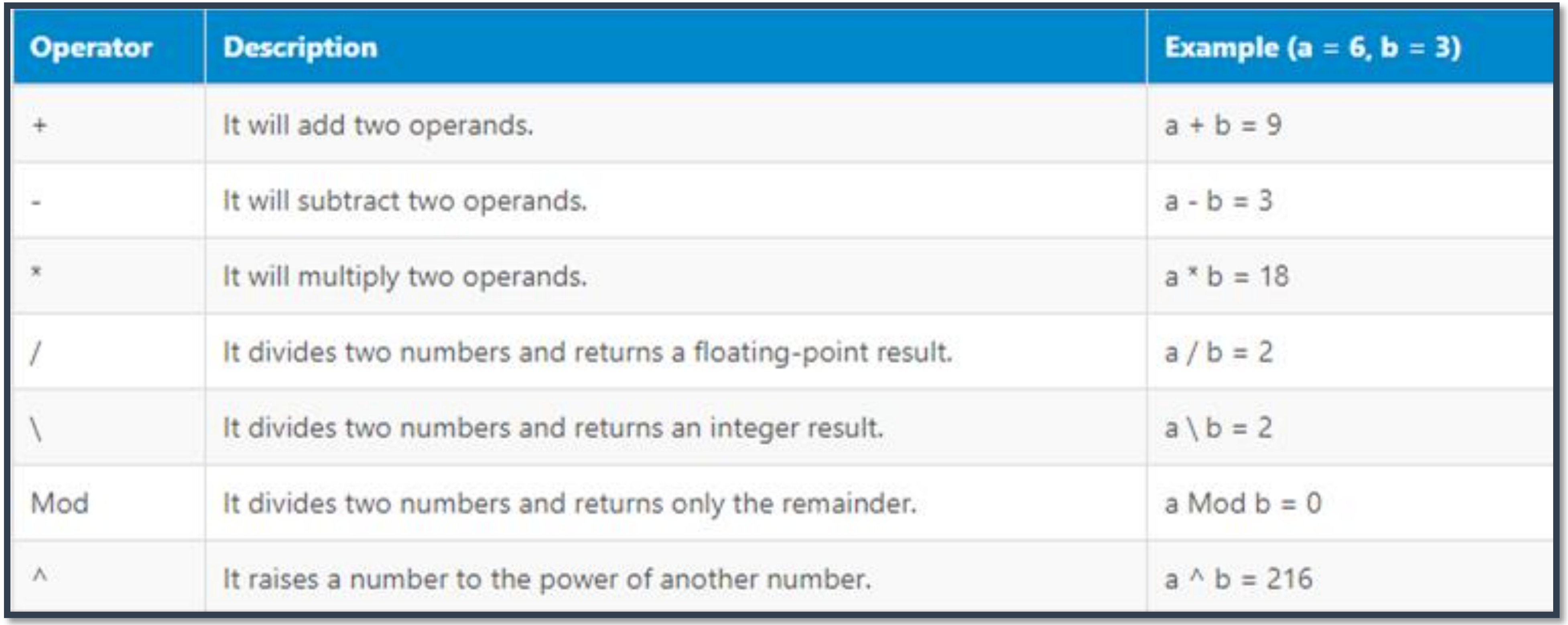
Reference Link –

https://www.javatpoint.com/vb-net-variable-and-constant

https://www.tutorialspoint.com/vb.net/vb.net_variables.htm

Arithmetic Operator in VB.Net

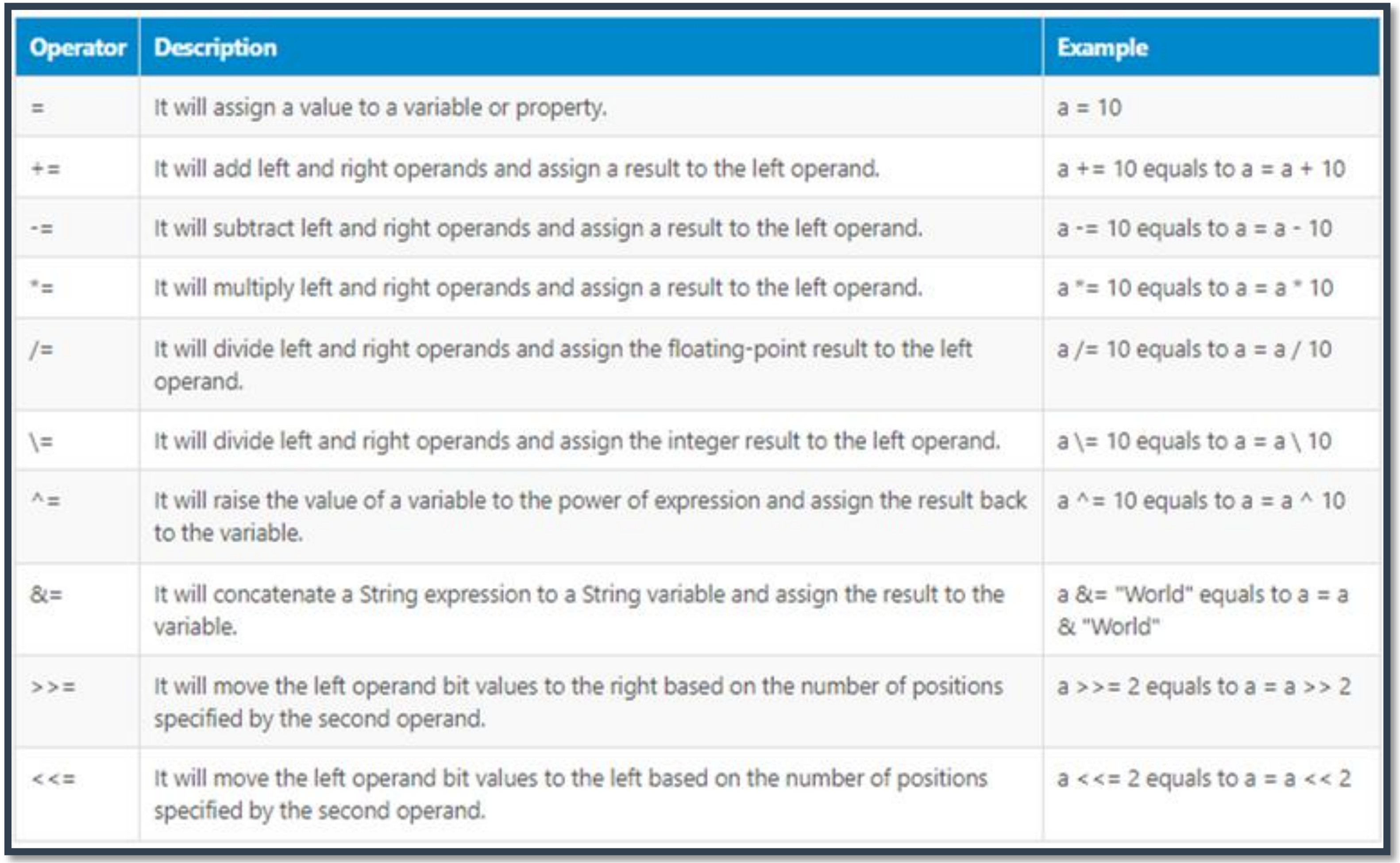
- **Arithmetic Operators** are useful for performing basic arithmetic calculations like addition, subtraction, division, etc., based on our requirements.



Operator	Description	Example (a = 6, b = 3)
+	It will add two operands.	a + b = 9
-	It will subtract two operands.	a - b = 3
*	It will multiply two operands.	a * b = 18
/	It divides two numbers and returns a floating-point result.	a / b = 2
\	It divides two numbers and returns an integer result.	a \ b = 2
Mod	It divides two numbers and returns only the remainder.	a Mod b = 0
^	It raises a number to the power of another number.	a ^ b = 216

Assignment Operator in VB.Net

- **Assignment Operators** are useful to assign a new value to the operand.

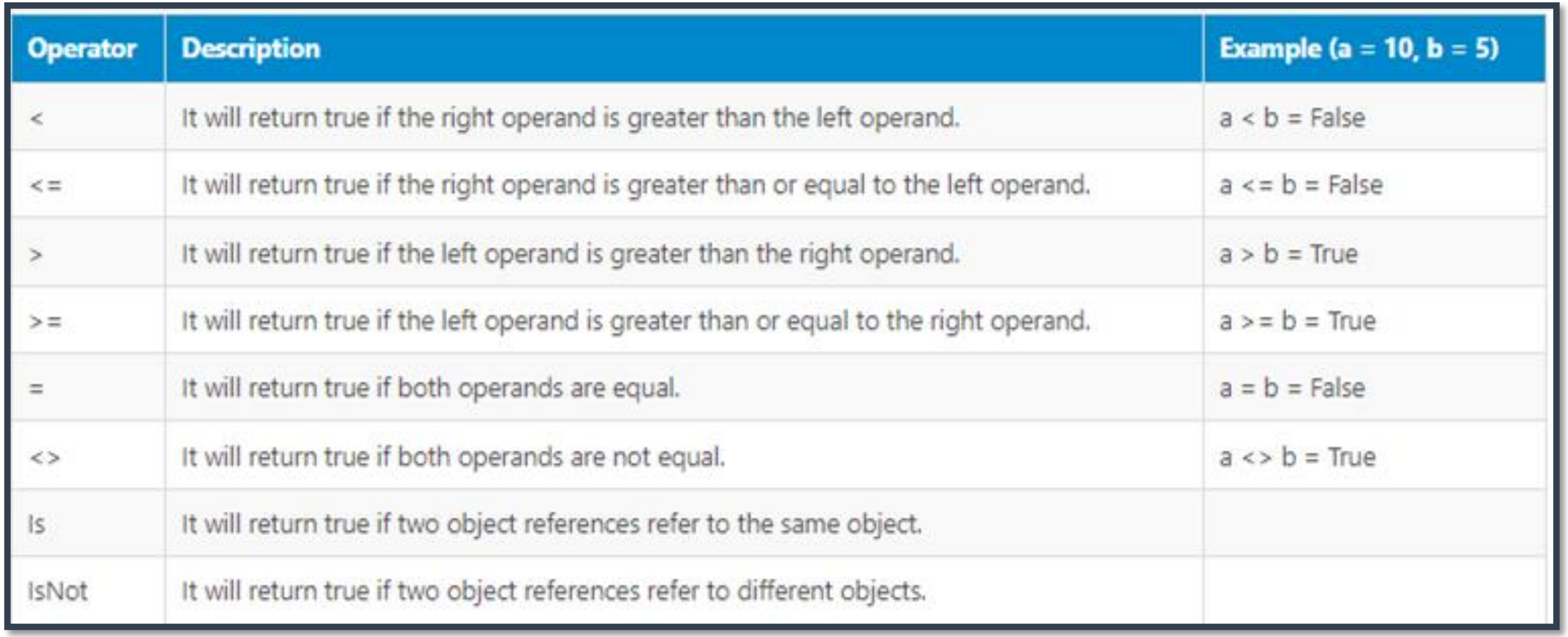


Operator	Description	Example
=	It will assign a value to a variable or property.	a = 10
+=	It will add left and right operands and assign a result to the left operand.	a += 10 equals to a = a + 10
-=	It will subtract left and right operands and assign a result to the left operand.	a -= 10 equals to a = a - 10
*=	It will multiply left and right operands and assign a result to the left operand.	a *= 10 equals to a = a * 10
/=	It will divide left and right operands and assign the floating-point result to the left operand.	a /= 10 equals to a = a / 10
\=	It will divide left and right operands and assign the integer result to the left operand.	a \= 10 equals to a = a \ 10
^=	It will raise the value of a variable to the power of expression and assign the result back to the variable.	a ^= 10 equals to a = a ^ 10
&=	It will concatenate a String expression to a String variable and assign the result to the variable.	a &= "World" equals to a = a & "World"
>>=	It will move the left operand bit values to the right based on the number of positions specified by the second operand.	a >>= 2 equals to a = a >> 2
<<=	It will move the left operand bit values to the left based on the number of positions specified by the second operand.	a <<= 2 equals to a = a << 2

Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

Comparison Operator in VB.Net

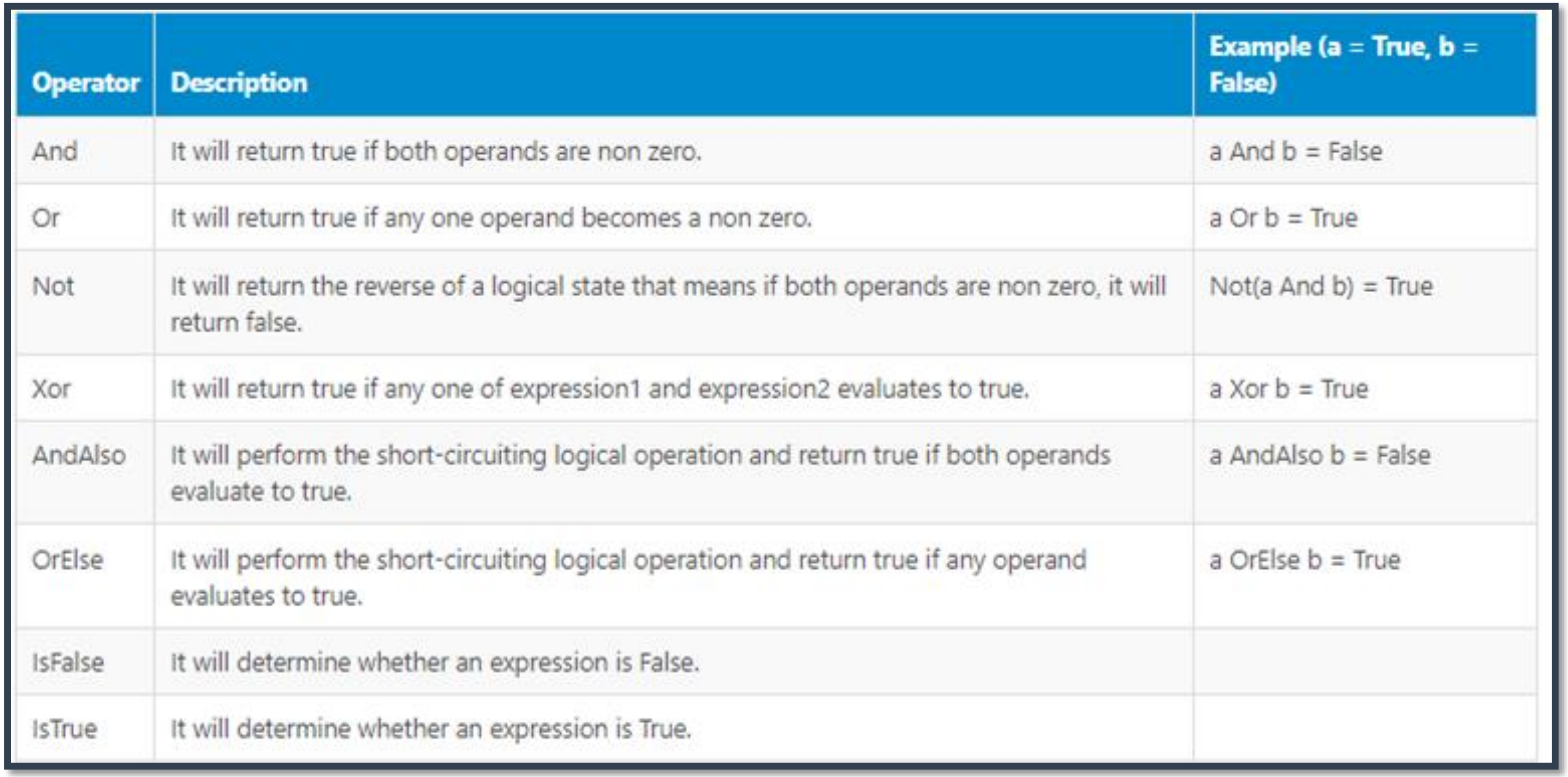
- **Comparison Operators** are useful to determine whether the defined two operands are equal, greater than or less than, etc., based on our requirements.



Operator	Description	Example (a = 10, b = 5)
<	It will return true if the right operand is greater than the left operand.	a < b = False
<=	It will return true if the right operand is greater than or equal to the left operand.	a <= b = False
>	It will return true if the left operand is greater than the right operand.	a > b = True
>=	It will return true if the left operand is greater than or equal to the right operand.	a >= b = True
=	It will return true if both operands are equal.	a = b = False
<>	It will return true if both operands are not equal.	a <> b = True
Is	It will return true if two object references refer to the same object.	
IsNot	It will return true if two object references refer to different objects.	

Logical / Bitwise Operator in VB.Net

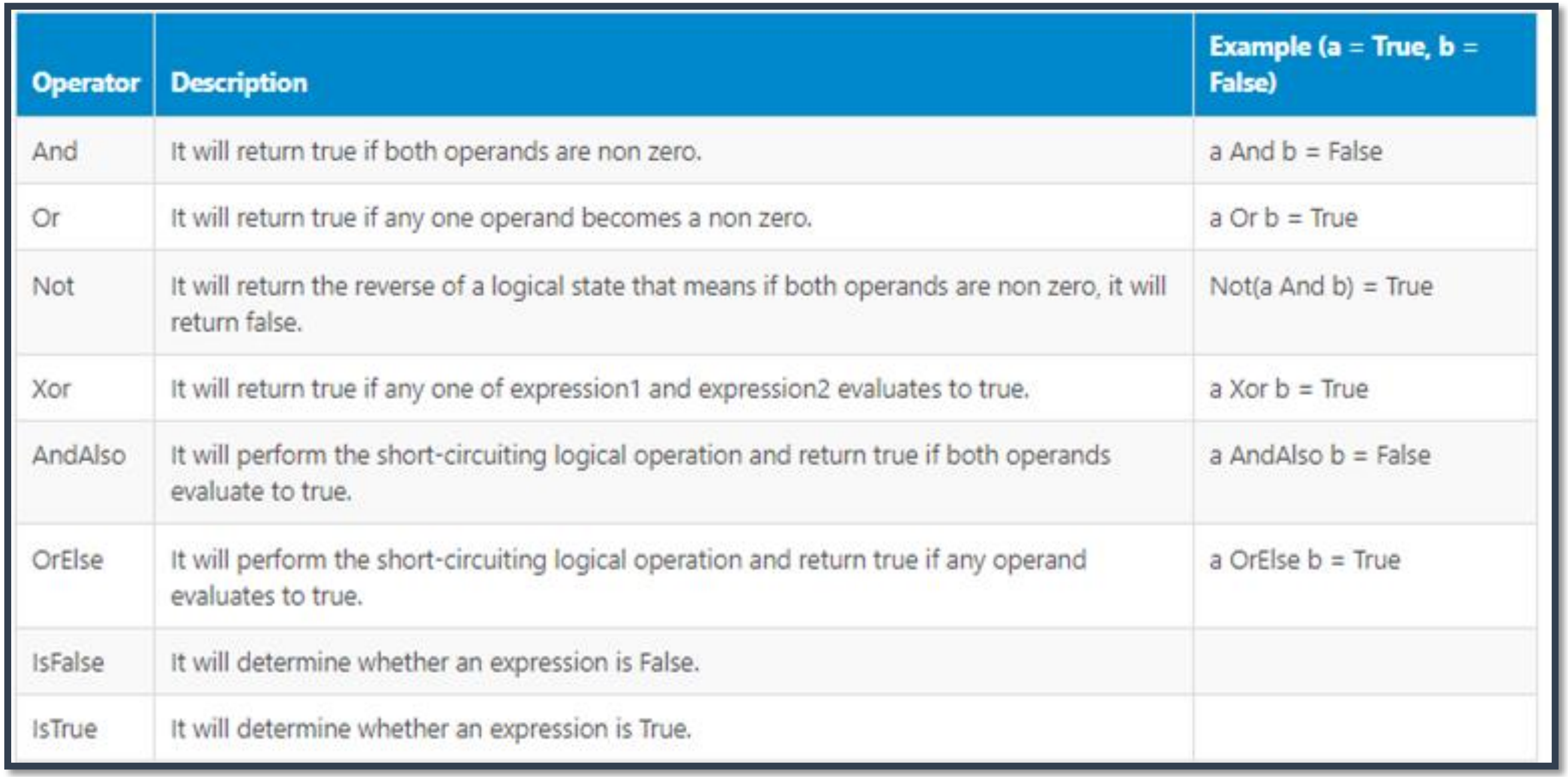
- **Logical / Bitwise** Operators are useful to perform the logical operation between two operands like AND, OR, etc., based on our requirements. The Logical / Bitwise Operators will always work with Boolean expressions (**true** or **false**) and return Boolean values.



Operator	Description	Example (a = True, b = False)
And	It will return true if both operands are non zero.	a And b = False
Or	It will return true if any one operand becomes a non zero.	a Or b = True
Not	It will return the reverse of a logical state that means if both operands are non zero, it will return false.	Not(a And b) = True
Xor	It will return true if any one of expression1 and expression2 evaluates to true.	a Xor b = True
AndAlso	It will perform the short-circuiting logical operation and return true if both operands evaluate to true.	a AndAlso b = False
OrElse	It will perform the short-circuiting logical operation and return true if any operand evaluates to true.	a OrElse b = True
IsFalse	It will determine whether an expression is False.	
IsTrue	It will determine whether an expression is True.	

Logical / Bitwise Operator in VB.Net

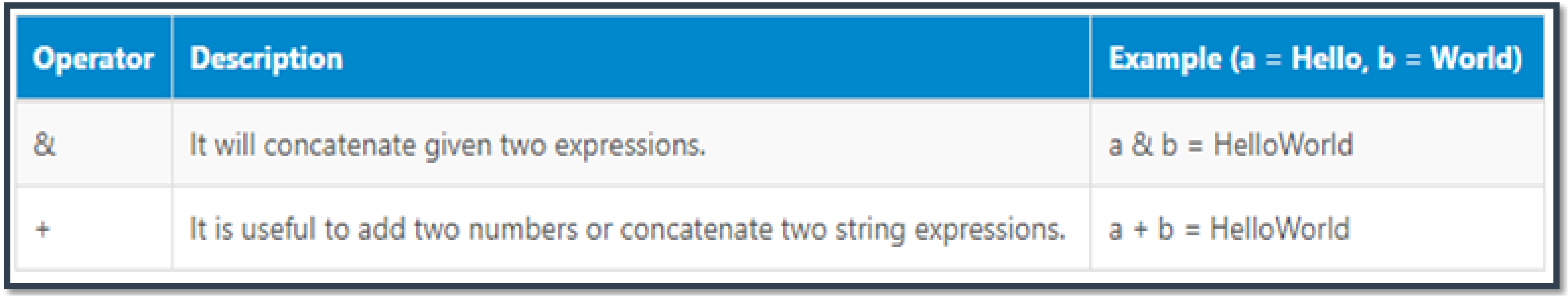
- **Logical / Bitwise** Operators are useful to perform the logical operation between two operands like AND, OR, etc., based on our requirements. The Logical / Bitwise Operators will always work with Boolean expressions (**true** or **false**) and return Boolean values.



Operator	Description	Example (a = True, b = False)
And	It will return true if both operands are non zero.	a And b = False
Or	It will return true if any one operand becomes a non zero.	a Or b = True
Not	It will return the reverse of a logical state that means if both operands are non zero, it will return false.	Not(a And b) = True
Xor	It will return true if any one of expression1 and expression2 evaluates to true.	a Xor b = True
AndAlso	It will perform the short-circuiting logical operation and return true if both operands evaluate to true.	a AndAlso b = False
OrElse	It will perform the short-circuiting logical operation and return true if any operand evaluates to true.	a OrElse b = True
IsFalse	It will determine whether an expression is False.	
IsTrue	It will determine whether an expression is True.	

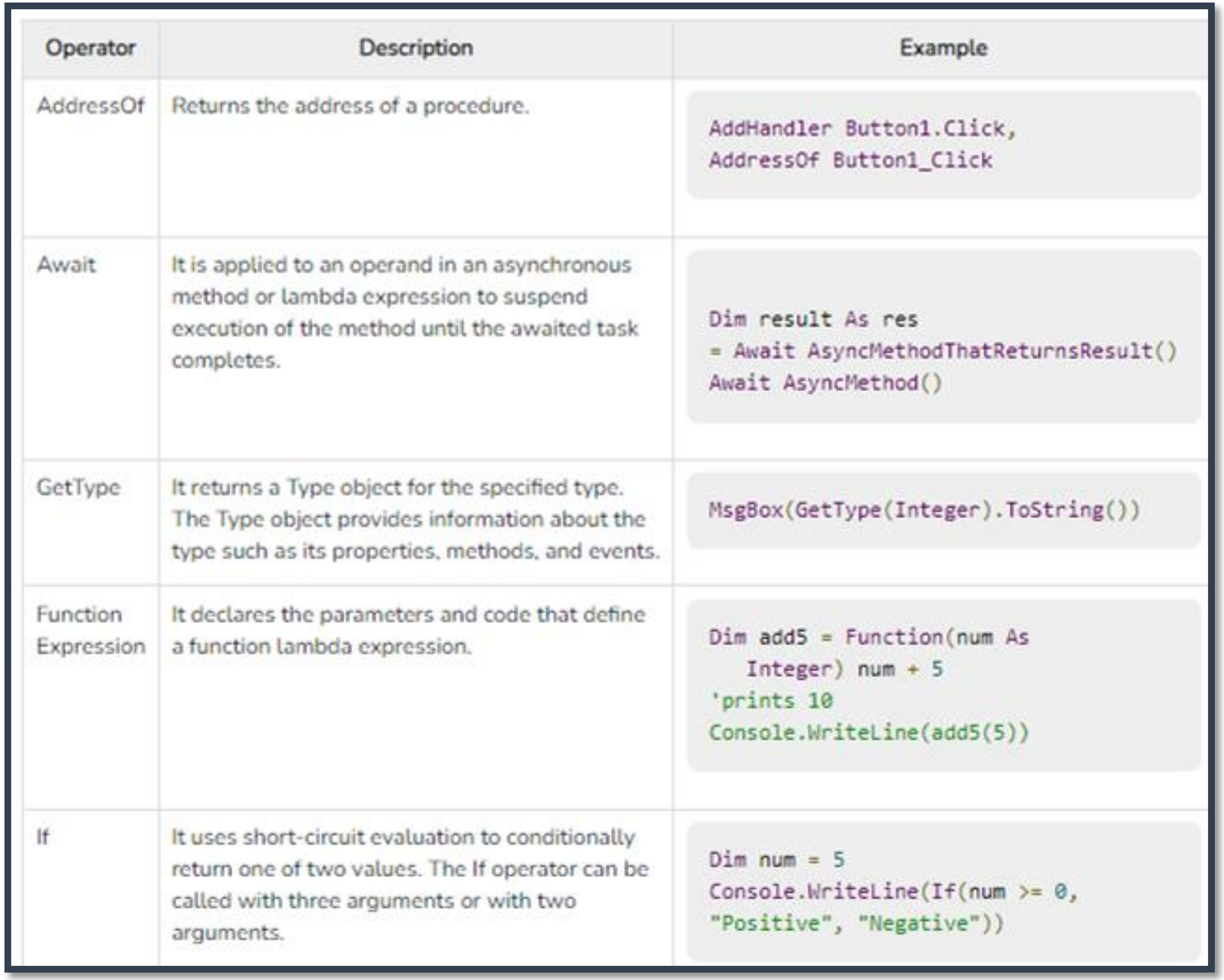
Concatenation Operator in VB.Net

- Concatenation Operators are useful to concatenate defined operands based on our requirements.



Operator	Description	Example (a = Hello, b = World)
&	It will concatenate given two expressions.	a & b = HelloWorld
+	It is useful to add two numbers or concatenate two string expressions.	a + b = HelloWorld

Miscellaneous Operator in VB.Net



Operator	Description	Example
AddressOf	Returns the address of a procedure.	AddHandler Button1.Click, AddressOf Button1_Click
Await	It is applied to an operand in an asynchronous method or lambda expression to suspend execution of the method until the awaited task completes.	Dim result As res = Await AsyncMethodThatReturnsResult() Await AsyncMethod()
GetType	It returns a Type object for the specified type. The Type object provides information about the type such as its properties, methods, and events.	MsgBox(GetType(Integer).ToString())

Function Expression	It declares the parameters and code that define a function lambda expression.	Dim add5 = Function(num As
		  Integer) num + 5
		'prints 10
		Console.WriteLine(add5(5))

If	It uses short-circuit evaluation to conditionally return one of two values. The If operator can be called with three arguments or with two arguments.	Dim num = 5 Console.WriteLine(If(num >= 0, "Positive", "Negative"))

Event and Event driven programming in VB.Net

- Visual Basic is an event-driven programming language. The event-driven programming is a computer programming paradigm where the flow and control of the program are determined by some events.
- In computer programming, the events are some user actions (such as mouse click, pressing a key, or hovering mouse) sensor outputs, messages or threads from other program code.
- In an event-driven programming, you can set to execute a block of program codes to when user mouse click, double click or even move the mouse.
- In visual basic, when the user action occurs for some events, then the particular block of codes that to be executed for that event is performed.

Basic Events in Visual Basic .NET

- An event is a signal or indicator to the .NET framework application that there has occurred some important action by user or program. The most basic events in Visual Basic .Net are given below:
- Click Event
- Load Event
- Double-Click Event
- Key Press Event
- Mouse Move Event

Event and Event driven programming in VB.Net

What is event driven programming?

- The event-driven programming revolves around recognizing the occurrences of events and then

responding to those events by taking appropriate actions.

- In event-driven programming an application is build up as a series of responses to user-events.

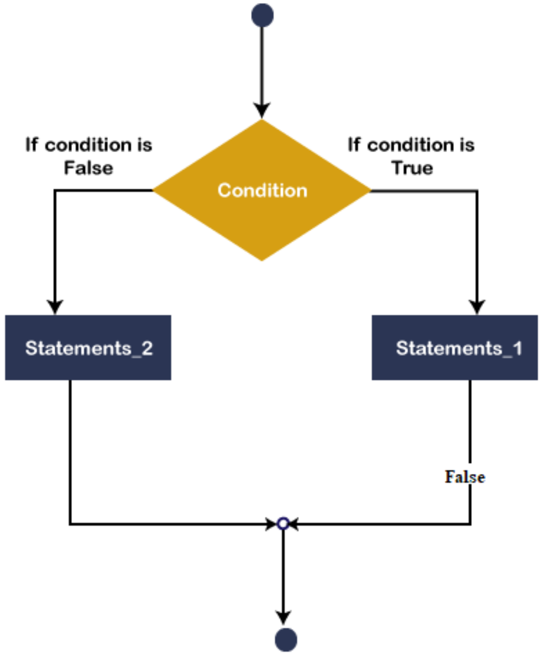
Why VB is called 'Event-Driven' programming language?

- In traditional or procedural application, the application itself determines which portion of code is to be executed and in what sequence. Generally execution starts with the 1st line of code and follow the coding sequence define in the application.
- Where as application written in VB are 'Event-Driven'. In an event-driven application the code doesn't follow a pre determined path rather it execute different code sections in response to events.
- Event can be triggered by user's action, by message from system, other applications or even from the application itself. The sequences of these events determine the order in which the code execute and associated with the objects of application. They either act on an object or are triggered by an object to control the flow of execution when it is running. That is why VB called Event-Driven programming language.

Control Statements in VB.Net

- Control statements are the statements that controls the execution of the program on the basis of the specified condition. It is useful for determining whether a condition is true or not. If the condition is true, a single or block of statement is executed. In the control statement, we will use if- Then, if Then Else, if Then Else If and the Select case statement.

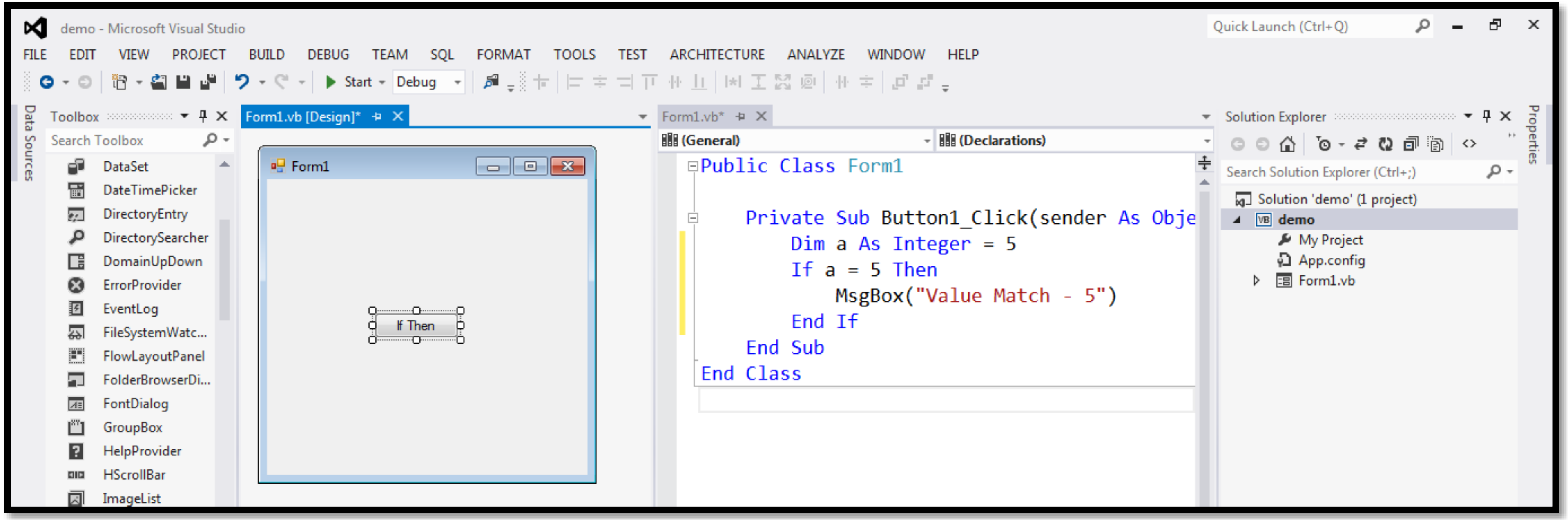
- **Control Statements in VB.NET**
  - **If-Then Statement**
  - **If-Then Else Statement**
  - **If-Then Else If Statement**
  - **Select Case Statement**
  - **Nested Select Case Statements**



If-Then Statement

- If...Then is the simplest form of control statement, frequently used in decision making and changing the control flow of the program execution.
- Syntax

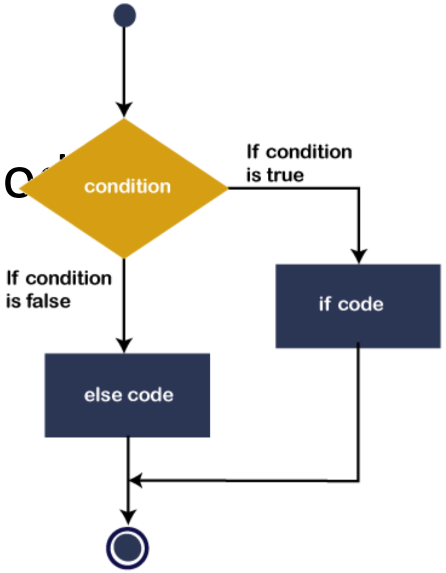
```
If condition Then
    [Statement(s)]
End If
```

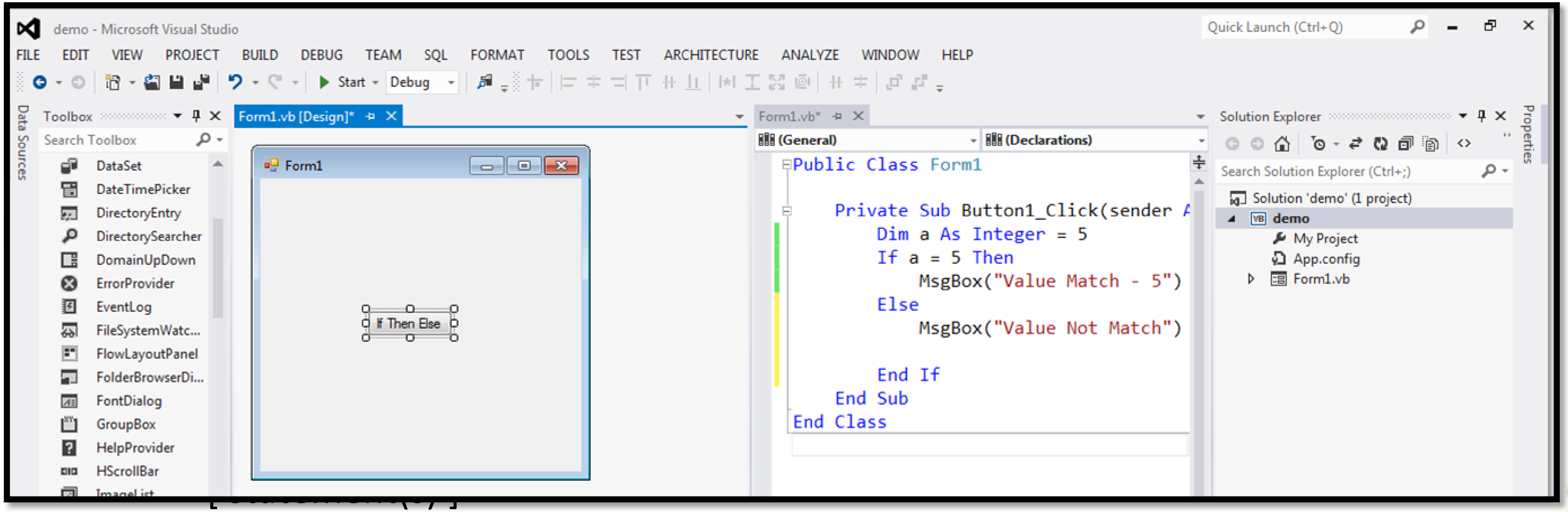


```
Public Class Form1

    Private Sub Button1_Click(sender As Obje
        Dim a As Integer = 5
        If a = 5 Then
            MsgBox("Value Match - 5")
        End If
    End Sub
End Class
```

If-Then Else Statement

- If...Then...Else statement, if the condition evaluates to true, the Bo the conditional statement runs. If the condition is false, the else-statement runs.





```
Public Class Form1

    Private Sub Button1_Click(sender A
        Dim a As Integer = 5
        If a = 5 Then
            MsgBox("Value Match - 5")
        Else
            MsgBox("Value Not Match")

        End If
    End Sub
End Class
```

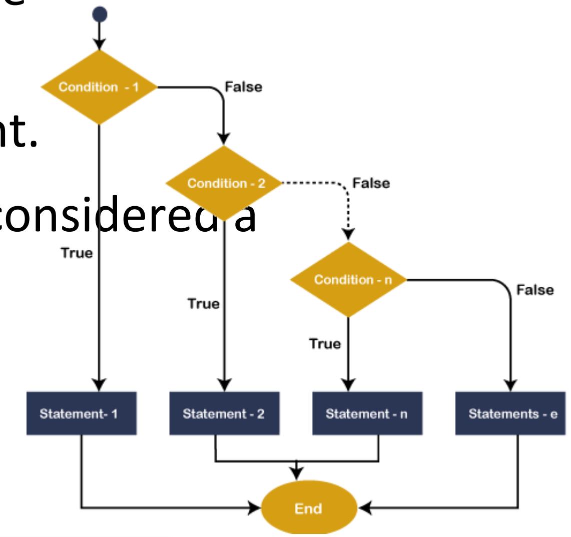
[statement(s)]

Else

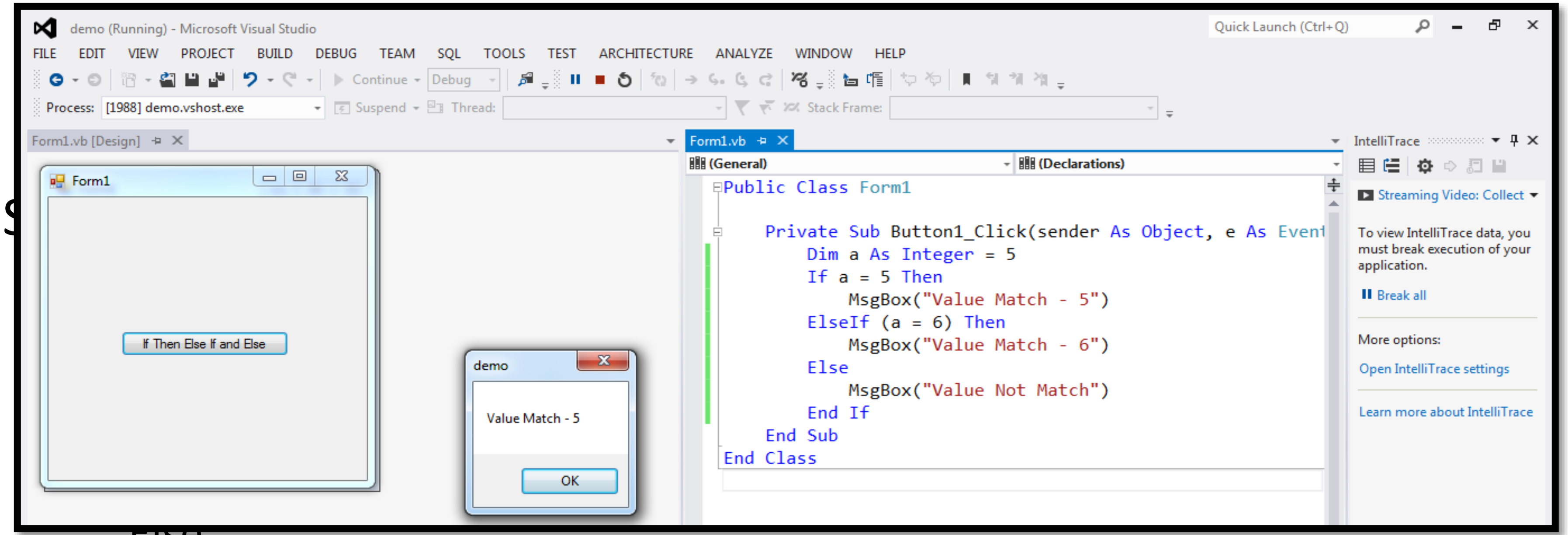
[elsestatement(s)]

Multiple If...Then...Else Statements

- In some cases, we need to use a sequence of If...Then structures or multiple If...Then...Else statements, where the Else clause is a new If structure.
- If we use nested If structures, the code would be pushed too far to the right.
- In such situations, it is allowed to use a new If right after the Else and it is considered a good practice.



- S



Process: [1988] demo.vshost.exe

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As Event
        Dim a As Integer = 5
        If a = 5 Then
            MsgBox("Value Match - 5")
        ElseIf (a = 6) Then
            MsgBox("Value Match - 6")
        Else
            MsgBox("Value Not Match")
        End If
    End Sub
End Class
```

Value Match - 5

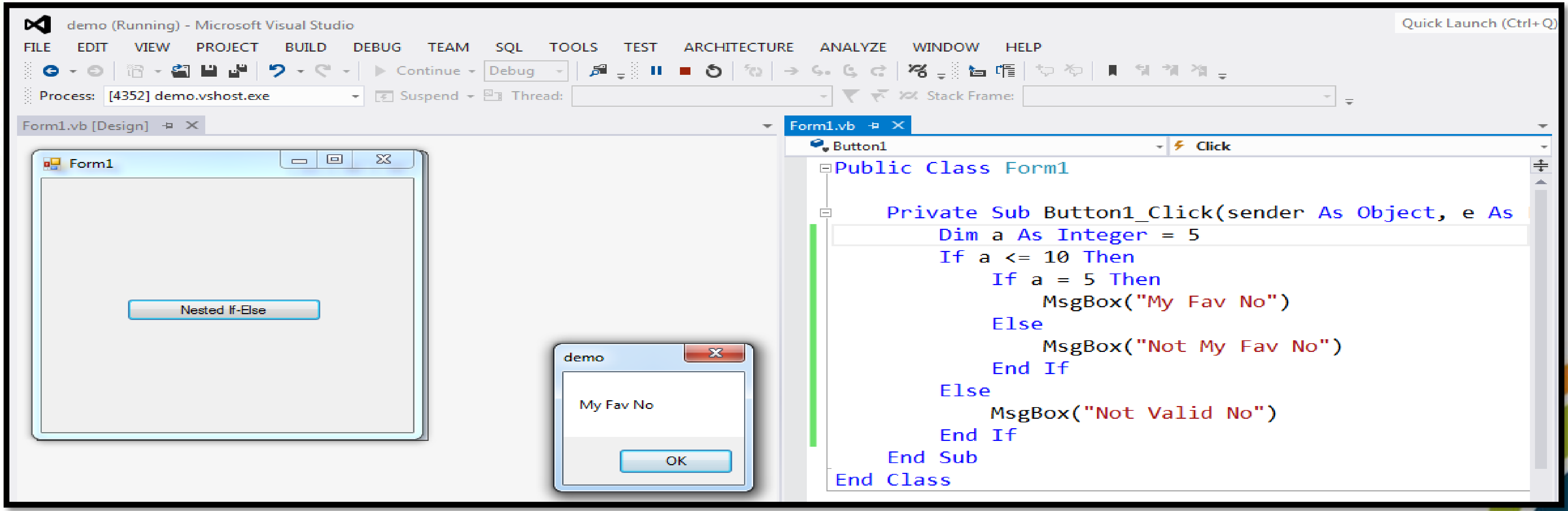
To view IntelliTrace data, you must break execution of your application.

More options:

Else

Nested If...Then...Else Statements

- Nested If-Else statements are useful to include one if…else statement within another if…else statement to test one condition followed by another condition.
- Generally, in Visual Basic, placing one if…else statement within another if…else statement is called a nested if…else statement.


demo (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Debug
Process: [4352] demo.vshost.exe
Suspend
Thread:
Stack Frame:
Form1.vb [Design]
Form1.vb
Button1
Click
Form1
Nested If-Else
demo
OK

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As
        Dim a As Integer = 5
        If a <= 10 Then
            If a = 5 Then
                MsgBox("My Fav No")
            Else
                MsgBox("Not My Fav No")
            End If
        Else
            MsgBox("Not Valid No")
        End If
    End Sub
End Class
```

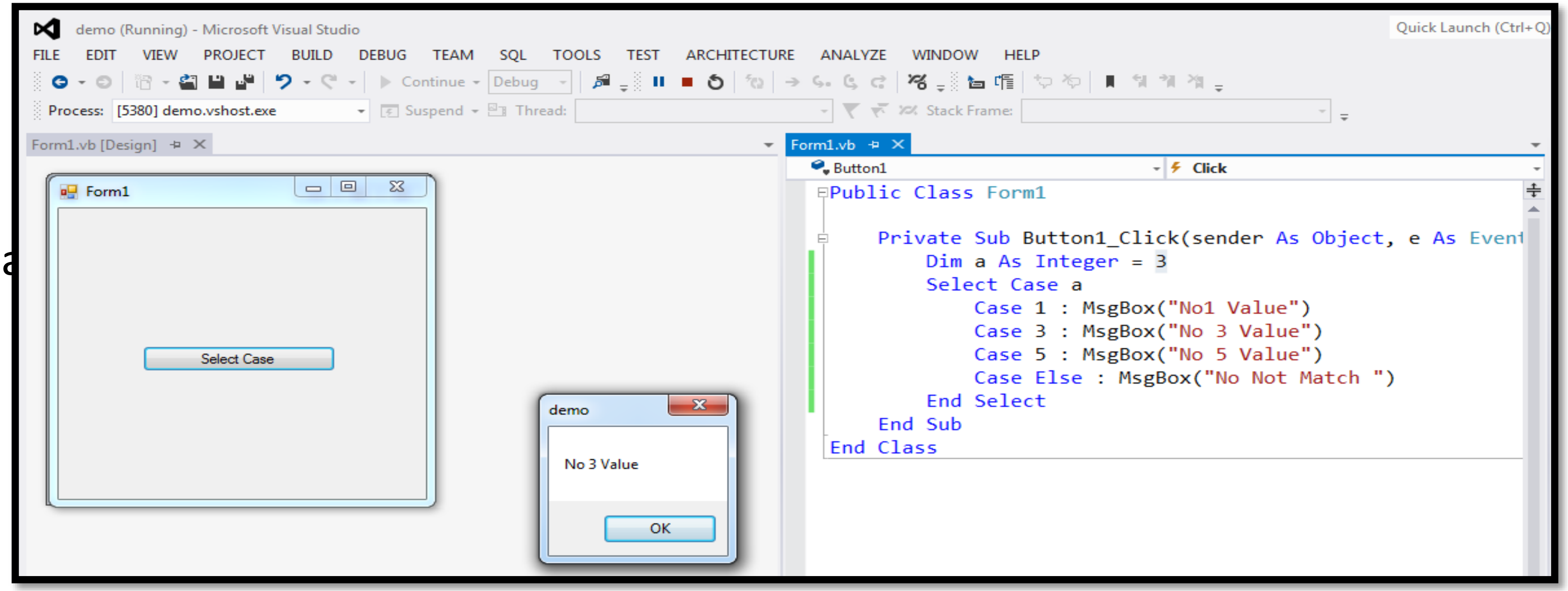
```
My Fav No
```

[statement(s)]

Select Case Statement

- Select Case statement is a collection of multiple case statements, which allows executing a single case statement from the list of statements. A selected case statement uses a variable to test for equality against multiple cases or statements in a program. If the variable is matched with any test cases, that statement will be executed. And if the condition is not matched with any cases, it executes the default statement.
- Using the select case statement in VB.NET programming, you can replace the uses of multiple If-Then-Else If statement from the program for better readability and easy to use.

- Synta


demo (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Debug
Process: [5380] demo.vshost.exe
Suspend
Thread:
Stack Frame:
Form1.vb [Design]
Form1.vb
Button1
Click
Form1
Select Case
demo
OK

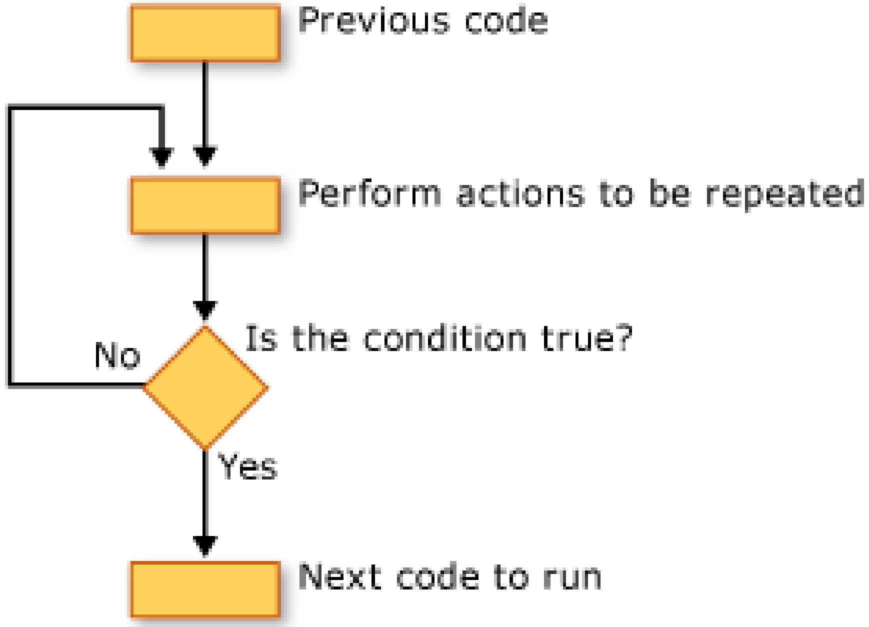
```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As Event
        Dim a As Integer = 3
        Select Case a
            Case 1 : MsgBox("No1 Value")
            Case 3 : MsgBox("No 3 Value")
            Case 5 : MsgBox("No 5 Value")
            Case Else : MsgBox("No Not Match ")
        End Select
    End Sub
End Class
```

No 3 Value

End Select

Looping in VB.Net

- Loop statements in programs are used to execute a block of program codes multiple times.
- Loop statements allow us to run a set of program codes which are written inside the loop for multiple times.
- The codes written inside the loop block are executed while only the given condition is true and the loop automatically ends when the condition false.
- **Looping Statements in VB.NET**
  - **Do Loop**
  - **For Next**
  - **For Each Next**
  - **While End While**
  - **With End With**



Do Loop Statement

- The Do Loop repeats the group of statements while or until the given Boolean condition is true. The Do Loop statements are terminated by the Exit Do statement.
- There are two methods of Do Loop. The first method is entry loop and the second method is exit do loop. In entry do loop the boolean condition is checks first, and the exit Do-loop checks the boolean condition after the execution of loop statements.

Syntax

Method – 1

```
Do {While | Until} condition
  [statement 1]
  [continue Do]
  [statement 2]
  [Exit Do]
  [statement X]
Loop
```

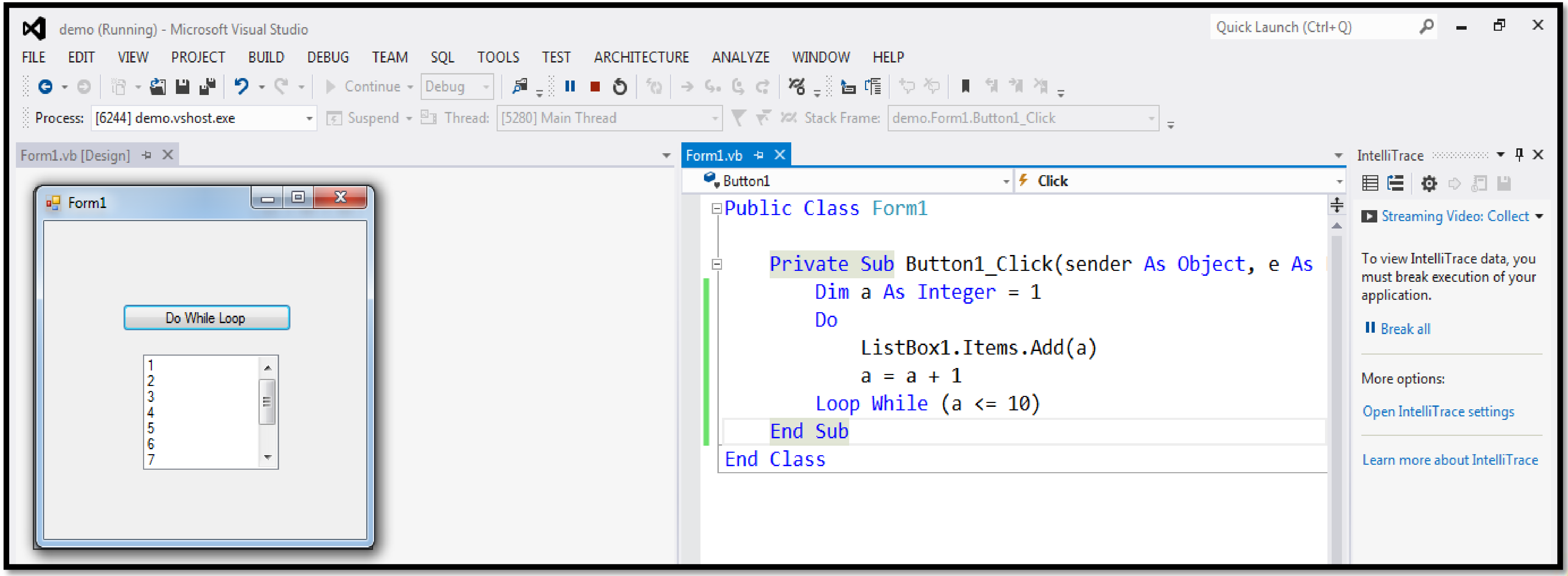
Method – 2

```
Do
  [statements 1]
  [Continue Do]
  [statements 2]
  [Exit Do]
  [Statements X]
Loop {While | Until} condition
```

Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

Do Loop Statement

Ex- Do Loop Method - 1

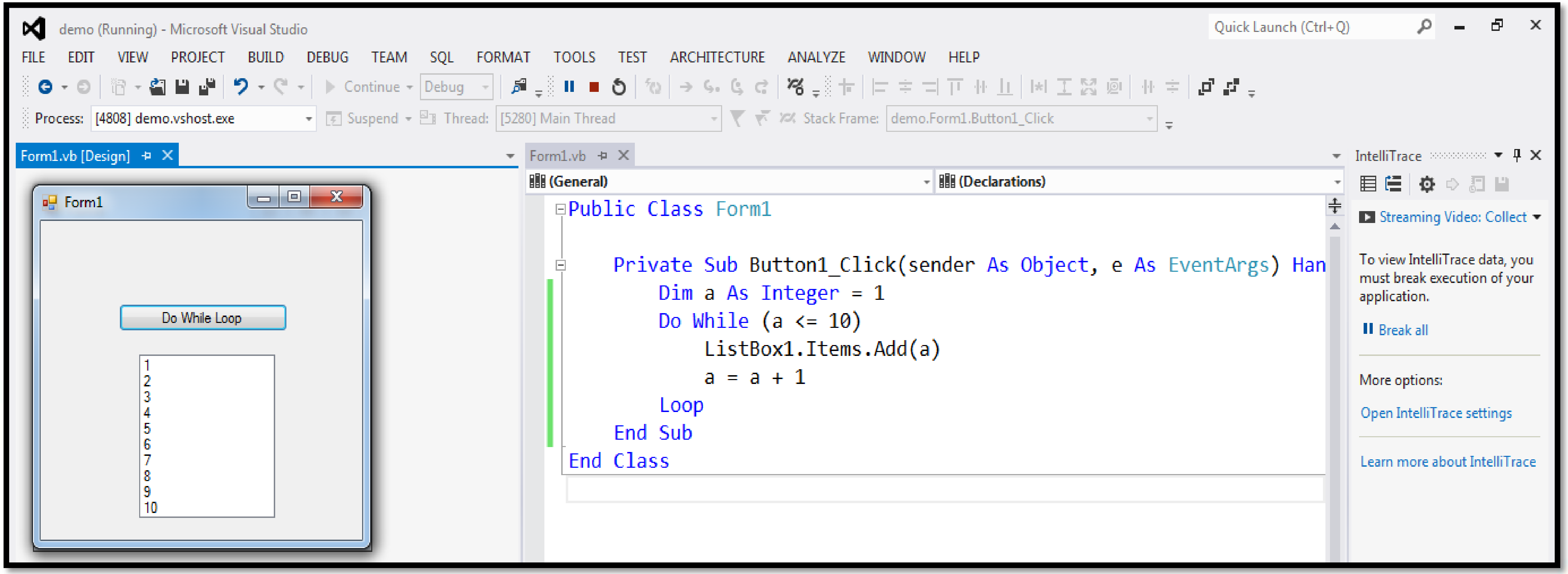

demo (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Debug
Process: [6244] demo.vshost.exe
Suspend
Thread: [5280] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1
Do While Loop
1
2
3
4
5
6
7
Form1.vb
Button1
Click
IntelliTrace
Streaming Video: Collect
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As
        Dim a As Integer = 1
        Do
            ListBox1.Items.Add(a)
            a = a + 1
        Loop While (a <= 10)
    End Sub
End Class
```

Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

Do Loop Statement

Ex- Do Loop Method - 2



demo (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL FORMAT TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Debug
Process: [4808] demo.vshost.exe
Suspend
Thread: [5280] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1.vb
(General)
(Declarations)
IntelliTrace
Streaming Video: Collect
Form1
Do While Loop
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace

```
1
2
3
4
5
6
7
8
9
10
```

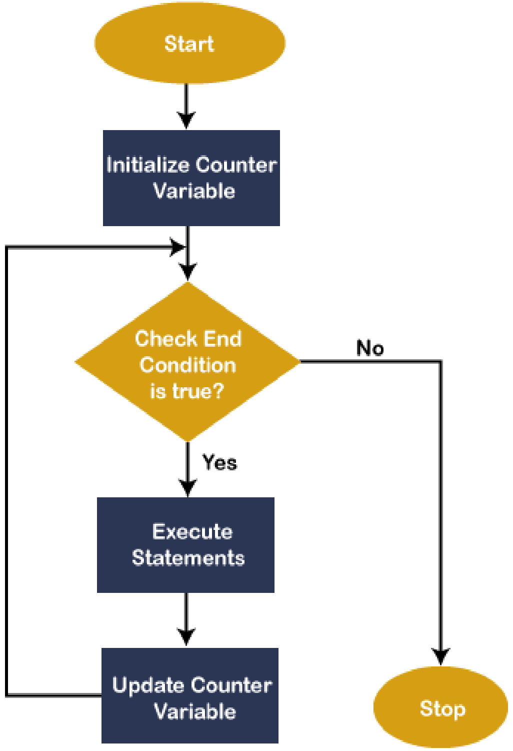
To view IntelliTrace data, you must break execution of your application.

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) Han
        Dim a As Integer = 1
        Do While (a <= 10)
            ListBox1.Items.Add(a)
            a = a + 1
        Loop
    End Sub
End Class
```

For Next Loop Statement

- A **For Next loop** is used to repeatedly execute a sequence of code or a block of code until a given condition is satisfied. A For loop is useful in such a case when we know how many times a block of code has to be executed. In VB.NET, the For loop is also known as For Next Loop.
- **Syntax**

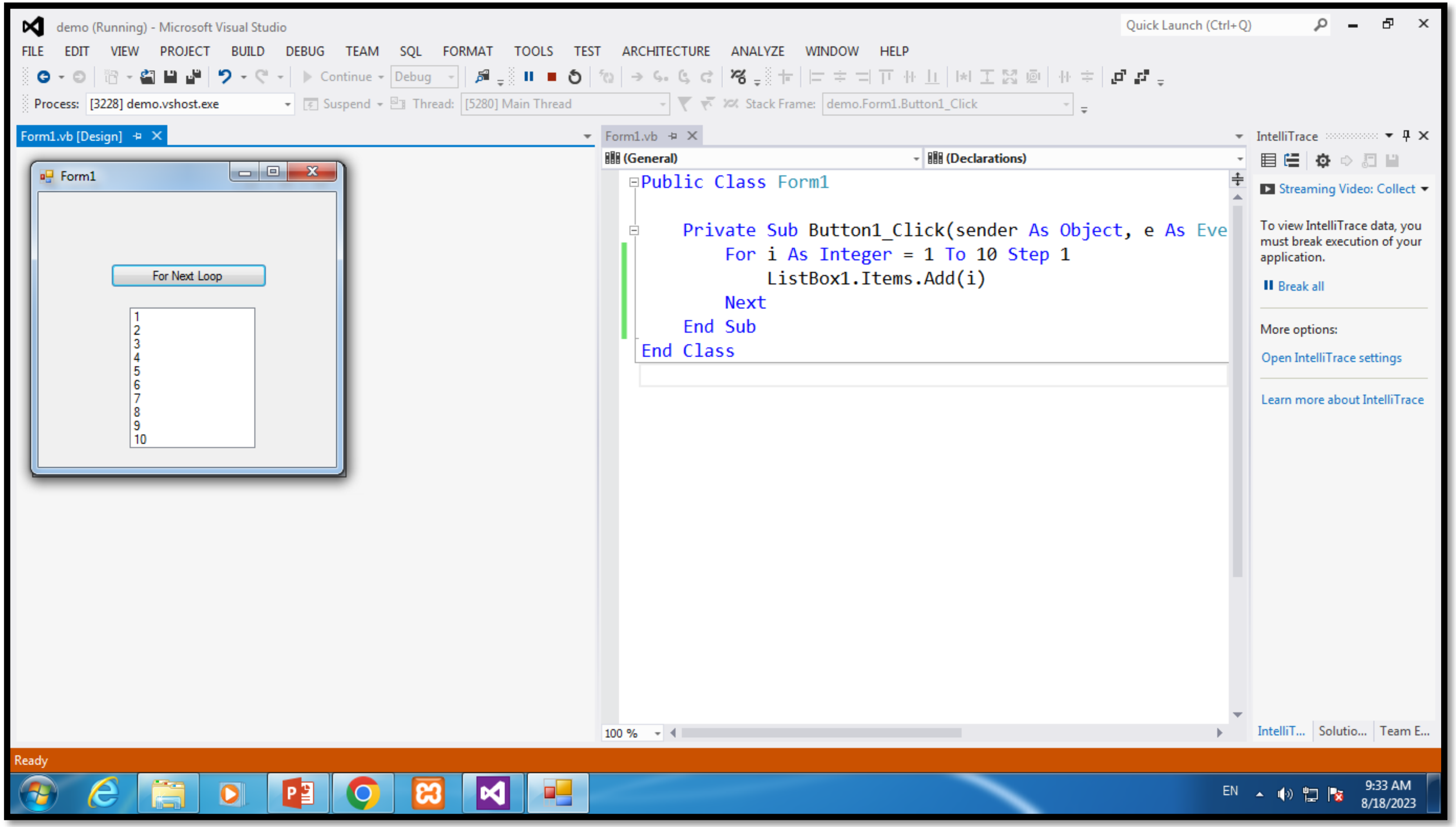
```
For variable_name As [ DataType ] = start To end [ Step step ]

          [ Statements to be executed ]

Next
```



Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

For Next Loop Statement

Ex-



demo (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL FORMAT TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [3228] demo.vshost.exe
Thread: [5280] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1.vb
Form1
For Next Loop
IntelliTrace
Streaming Video: Collect
Break all
Open IntelliTrace settings
Learn more about IntelliTrace
Ready

```
1
2
3
4
5
6
7
8
9
10
```

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As Eve
        For i As Integer = 1 To 10 Step 1
            ListBox1.Items.Add(i)
        Next
    End Sub
End Class
```

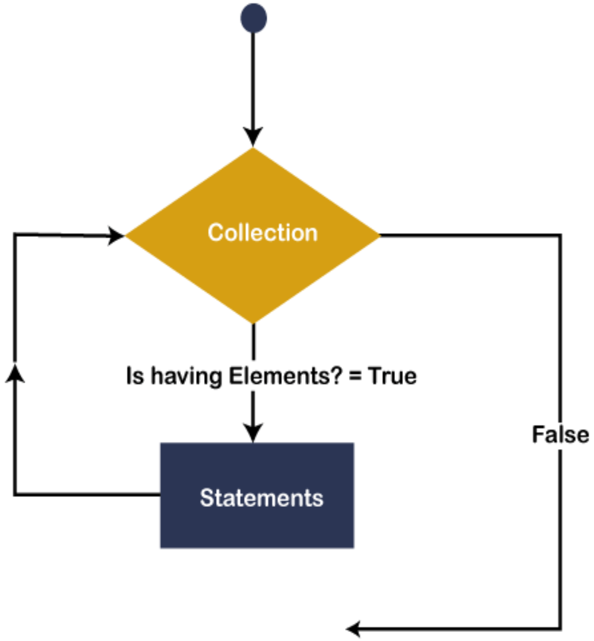
To view IntelliTrace data, you must break execution of your application.

More options:

For Each Loop Statement

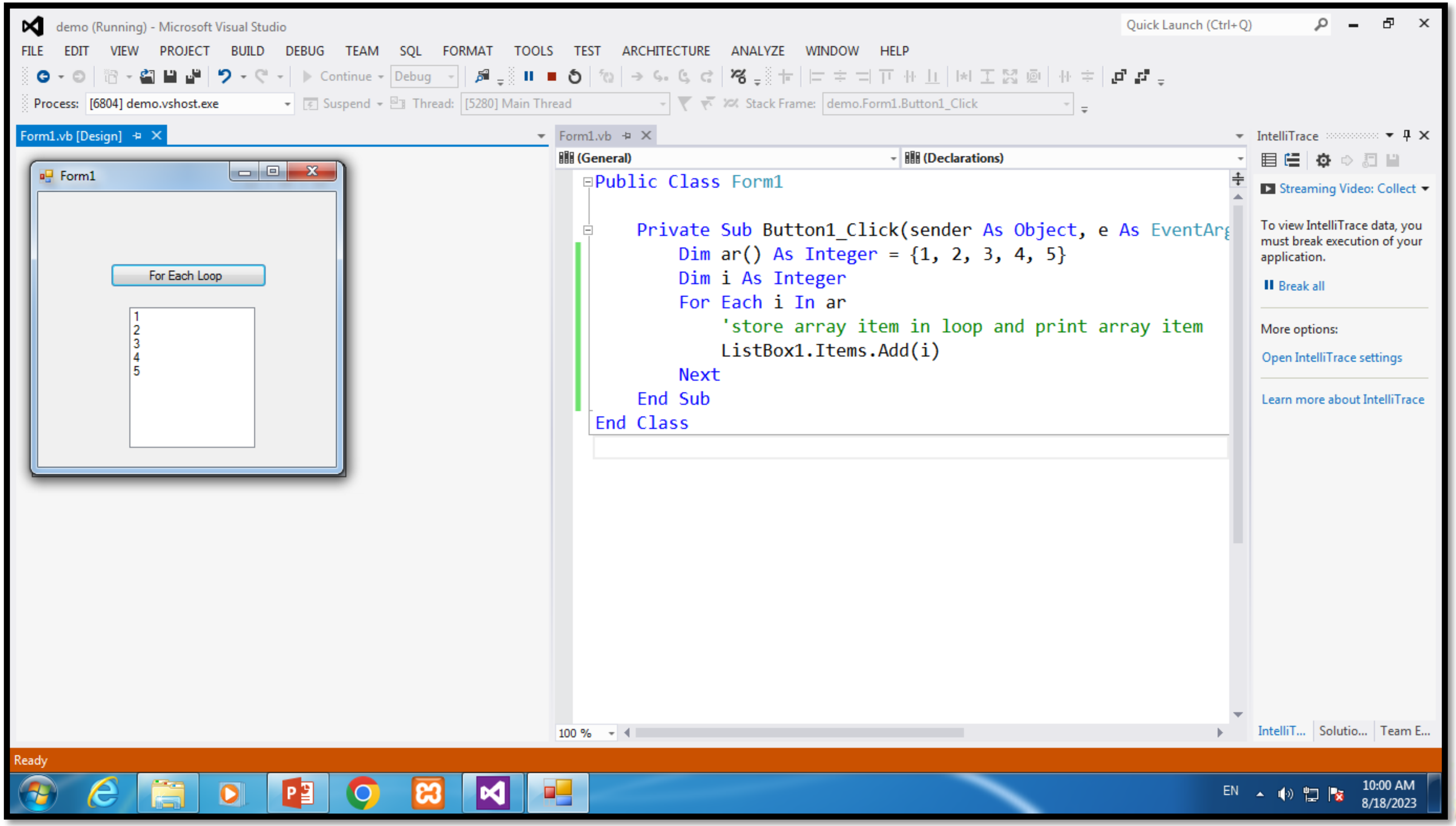
- For Each loop is used to iterate block of statements in an array or collection objects. Using For Each loop, we can easily work with collection objects such as lists, arrays, etc., to execute each element of an array or in a collection. And when iteration through each element in the array or collection is complete, the control transferred to the next statement to end the loop.
- **Syntax**

```
For Each var_name As [ DataType ] In Collection_Object

        [ Statements to be executed]

Next
```



For Each Loop Statement

Ex-



demo (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL FORMAT TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [6804] demo.vshost.exe
Thread: [5280] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1
For Each Loop
Form1.vb
(General)
(Declarations)
IntelliTrace
Streaming Video: Collect
Break all
Open IntelliTrace settings
Learn more about IntelliTrace
Ready

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArg
        Dim ar() As Integer = {1, 2, 3, 4, 5}
        Dim i As Integer
        For Each i In ar
            'store array item in loop and print array item
            ListBox1.Items.Add(i)
        Next
    End Sub
End Class
```

```
1
2
3
4
5
```

To view IntelliTrace data, you must break execution of your application.

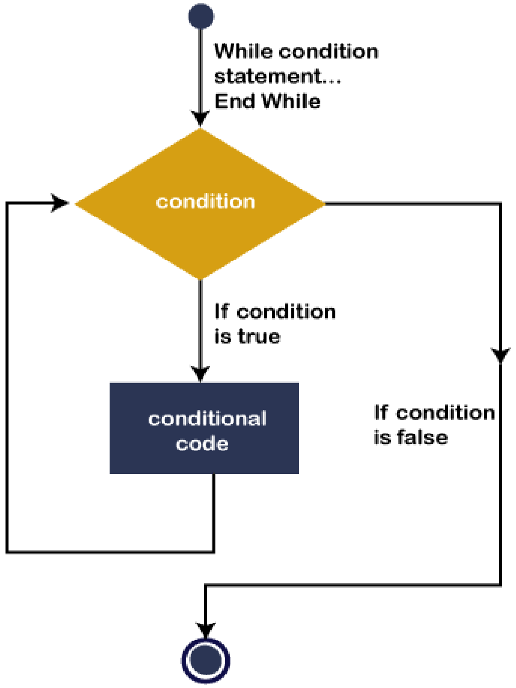
More options:

Design and Develop by Assistant Professor Ankit Rami **Any Query Contact - 8460467193**

While End Loop Statement

- **While End loop** is used to execute blocks of code or statements in a program, as long as the given **condition** is true. It is useful when the number of executions of a block is not known. It is also known as an **entry-controlled loop** statement, which means it initially checks all loop conditions. If the condition is true, the body of the while loop is executed. This process of repeated execution of the body continues until the condition is not false. And if the condition is false, control is transferred out of the loop.
- **condition** represents any **Boolean condition,** and if the logical condition is true, the **single or block of statements** define inside the body of the while loop is executed.
- **Syntax**

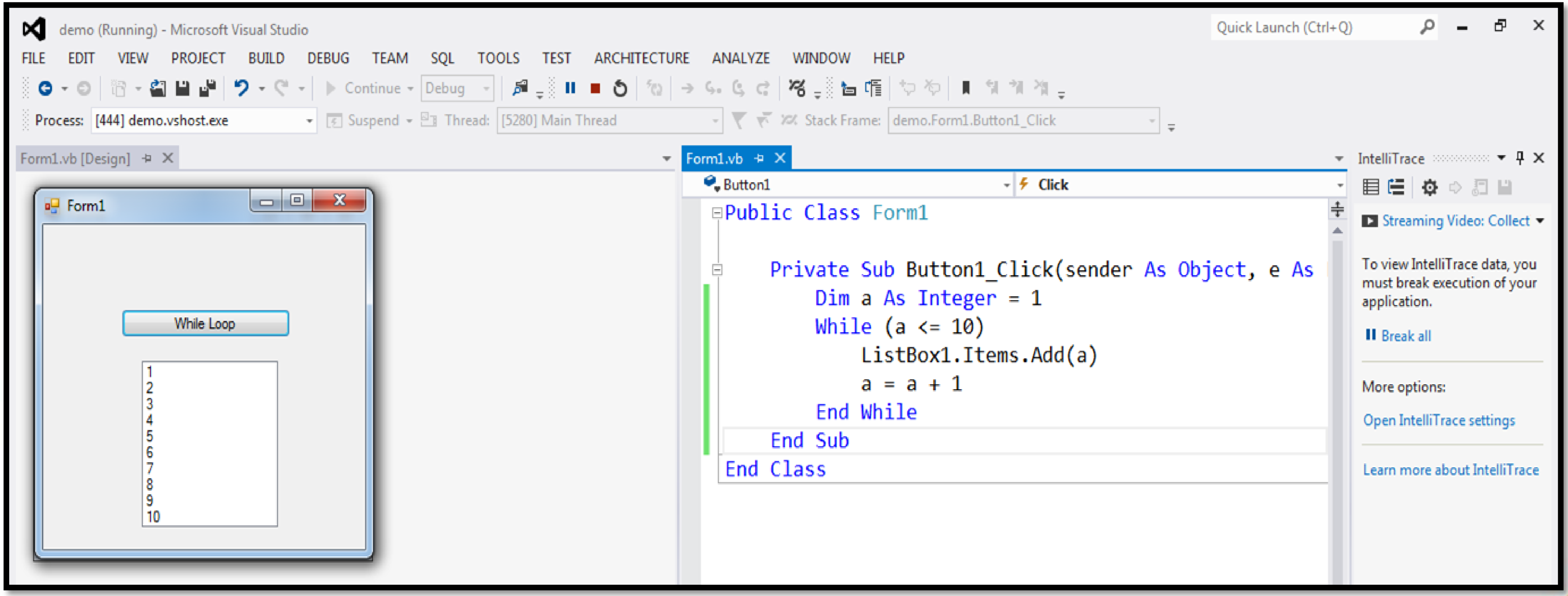
```
While [condition]

    [ Statement to be executed ]

End While
```



Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

While End Loop Statement

Ex-


demo (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue Debug
Process: [444] demo.vshost.exe
Suspend Thread: [5280] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1
While Loop
1
2
3
4
5
6
7
8
9
10
Form1.vb
Button1
Click
IntelliTrace
Streaming Video: Collect
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace

```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As
        Dim a As Integer = 1
        While (a <= 10)
            ListBox1.Items.Add(a)
            a = a + 1
        End While
    End Sub
End Class
```

With End With Loop Statement

- **With End** statement is not the same as a loop structure. It is used to access and execute statements on a specified object without specifying the name of the objects with each statement. Within a **With** statement block, you can specify a member of an object that begins with a period **(.)** to define multiple statements.
- **Syntax**

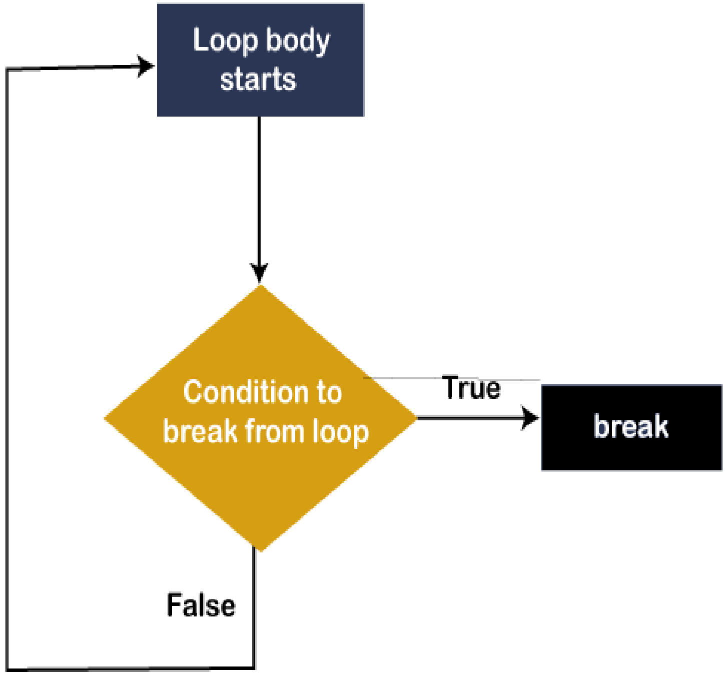
```
With objExpression

   [ Statements to be Executed]

End With
```

Exit Statement in VB.Net

- Exit statement is used to terminate the loop (for, while, do, select case, etc.) or exit the loop and pass control immediately to the next statement of the termination loop. Furthermore, the Exit statement can also be used in the nested loop to stop or terminate the execution of the inner or outer loop at any time, depending on our requirements.

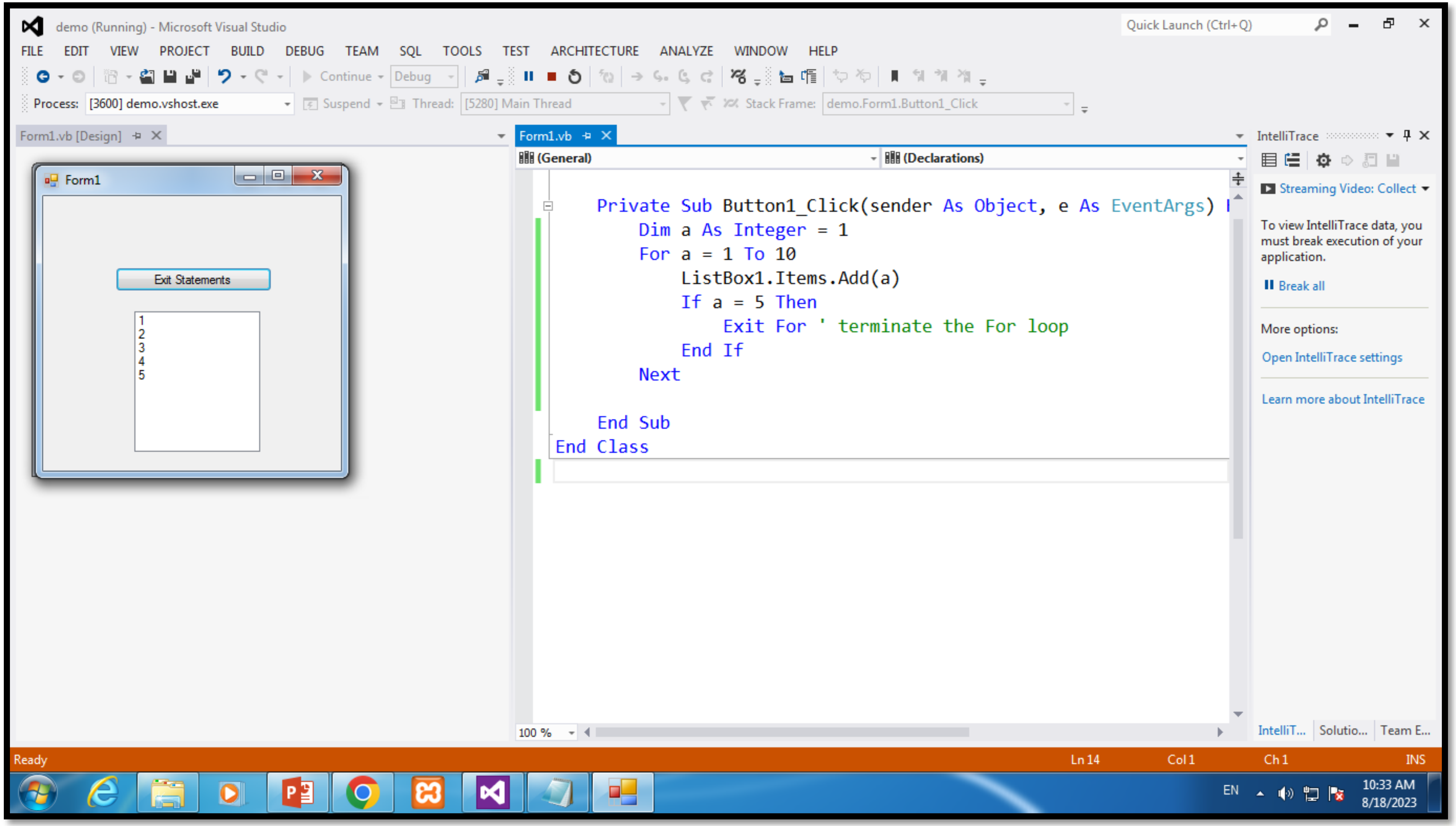
- **Syntax**

```
Exit { Do | For | Function | Property | Select | Sub | Try | While }
```



Exit Statement in VB.Net

Ex-


demo (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Continue
Debug
Process: [3600] demo.vshost.exe
Suspend
Thread: [5280] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1.vb
(General)
(Declarations)
Form1
Exit Statements
IntelliTrace
Streaming Video: Collect
To view IntelliTrace data, you must break execution of your application.
Break all
More options:
Open IntelliTrace settings
Learn more about IntelliTrace
100 %
IntelliT... Solutio... Team E...
Ready
Ln 14
Col 1
Ch 1
INS
EN
10:33 AM
8/18/2023

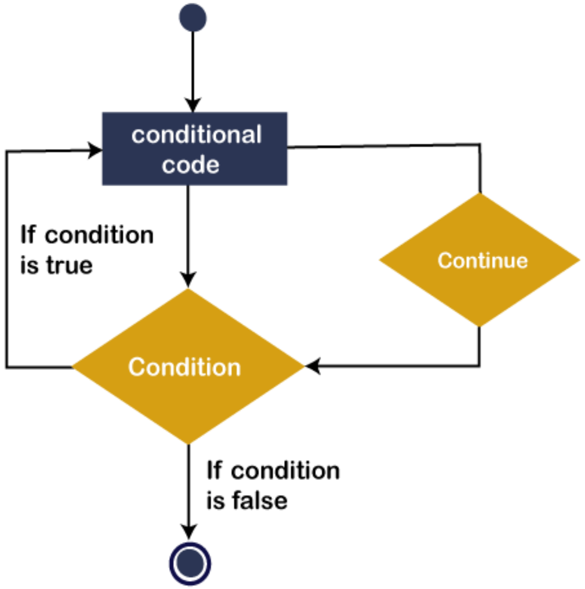
```
    Private Sub Button1_Click(sender As Object, e As EventArgs)
        Dim a As Integer = 1
        For a = 1 To 10
            ListBox1.Items.Add(a)
            If a = 5 Then
                Exit For ' terminate the For loop
            End If
        Next

    End Sub
End Class
```

```
1
2
3
4
5
```

Continue Statement in VB.Net

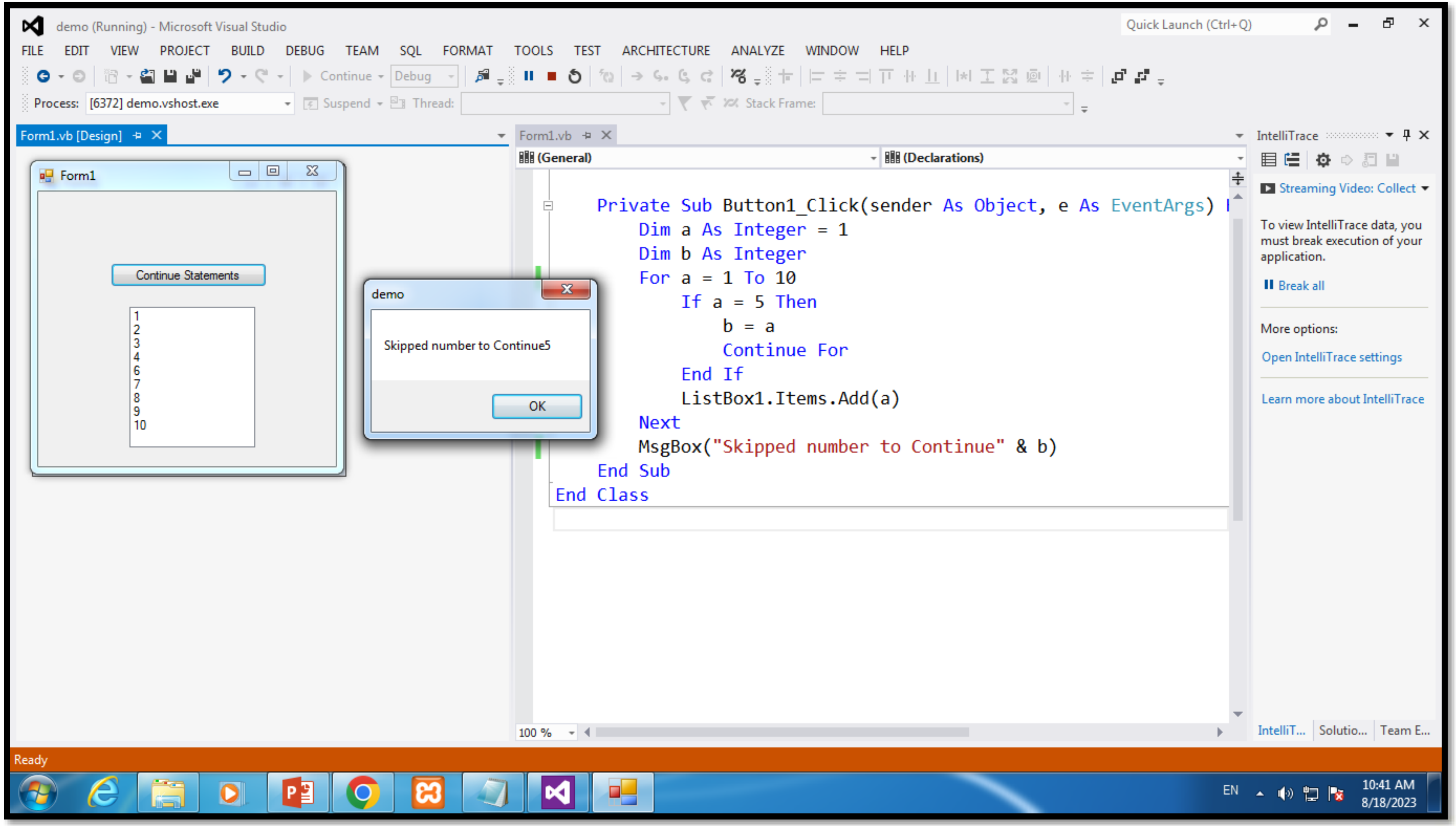
- **continue** statement is used to skip the particular iteration of the loop and continue with the next iteration. Generally, the **continue** Statement is written inside the body of the For, While and Do While loop with a condition. In the previous section, we learned about the Exit Statement. The main difference between the **Exit** and a **Continue** Statement is that the Exit Statement is used to exit or terminate the loop's execution process. In contrast, the Continue Statement is used to **Skip** the particular iteration and **continue** with the next iteration **without** ending the loop.
- **Syntax**

```
Continue { Do | For | While }
```



Continue Statement in VB.Net

Ex-



demo (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL FORMAT TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [6372] demo.vshost.exe
Form1.vb [Design]
Form1.vb
(General)
(Declarations)
Form1
Continue Statements
IntelliTrace
Ready

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) H
        Dim a As Integer = 1
        Dim b As Integer
        For a = 1 To 10
            If a = 5 Then
                b = a
                Continue For
            End If
            ListBox1.Items.Add(a)
        Next
        MsgBox("Skipped number to Continue" & b)
    End Sub
End Class
```

```
1
2
3
4
6
7
8
9
10
```

demo

Skipped number to Continue5

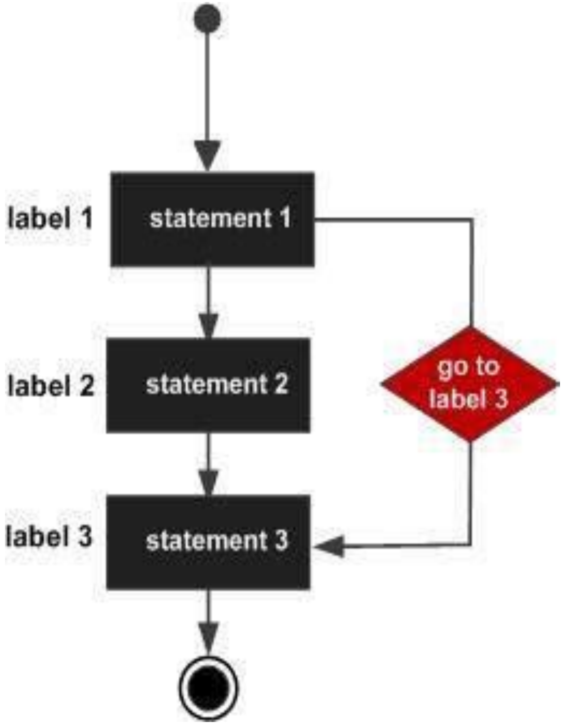
OK

Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

GoTo Statement in VB.Net

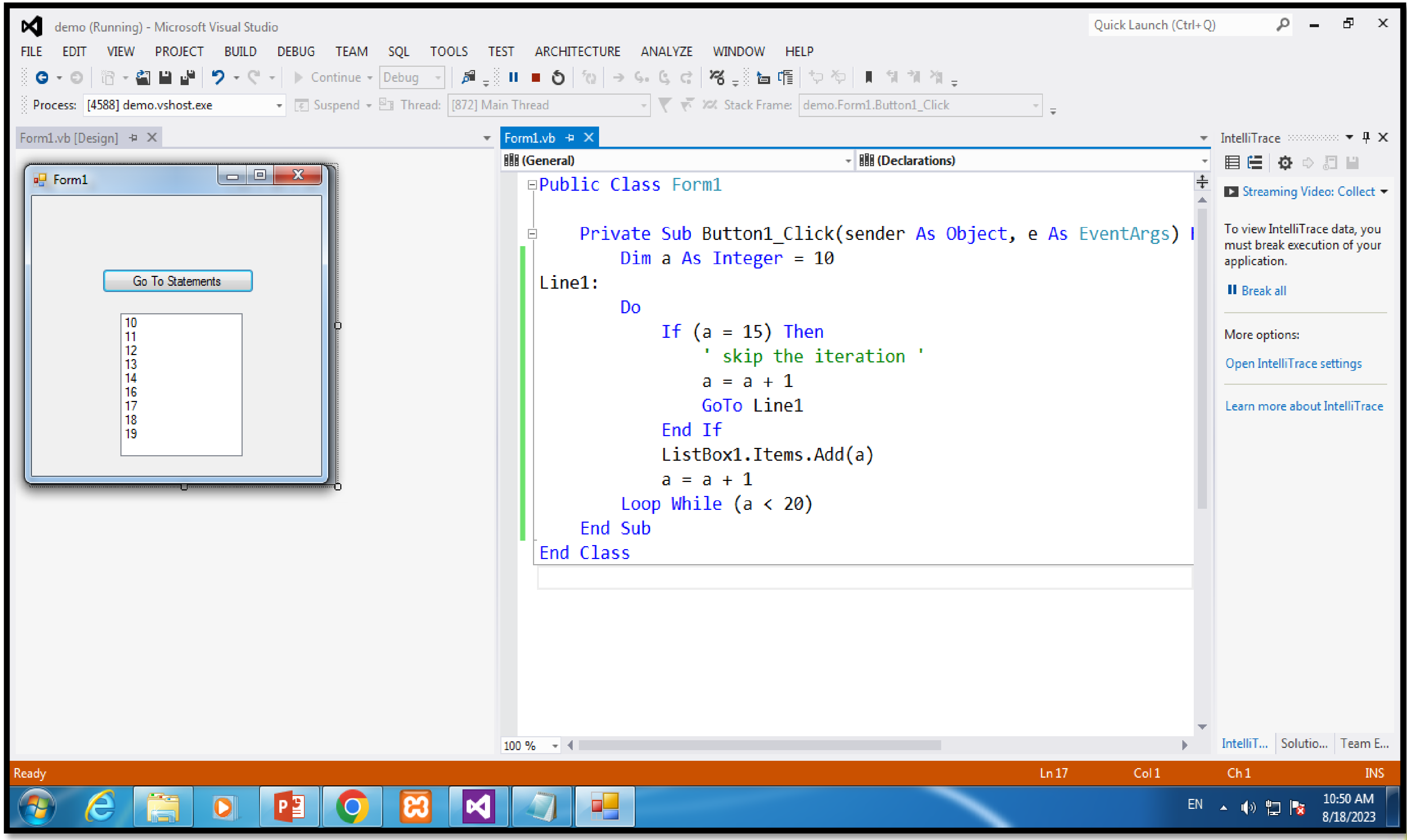
- **GoTo** statement is known as a jump statement. It is a control statement that transfers the flow of control to the specified label within the procedure. The GoTo statement uses labels that must be a valid identifier. The GoTo statement can be used in Select case, decision control statements, and loops.
- **Syntax**

GoTo label_1



GoTo Statement in VB.Net

Ex-



demo (Running) - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Process: [4588] demo.vshost.exe
Thread: [872] Main Thread
Stack Frame: demo.Form1.Button1_Click
Form1.vb [Design]
Form1.vb
(General)
(Declarations)
Form1
Go To Statements
IntelliTrace
Ready
Ln 17
Col 1
Ch 1
INS

```
10
11
12
13
14
16
17
18
19
```

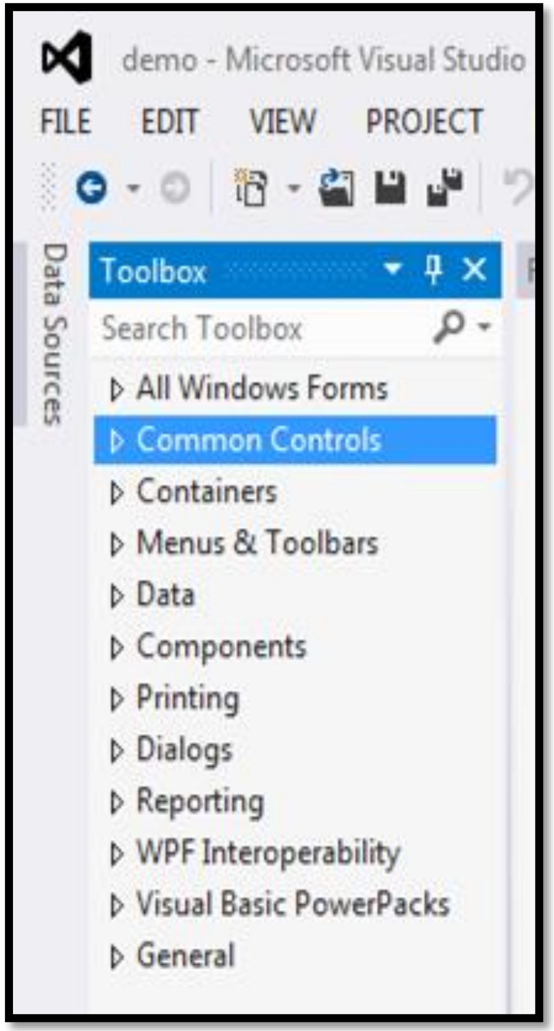
```
Public Class Form1

    Private Sub Button1_Click(sender As Object, e As EventArgs) I
        Dim a As Integer = 10
Line1:
        Do
            If (a = 15) Then
                ' skip the iteration '
                a = a + 1
                GoTo Line1
            End If
            ListBox1.Items.Add(a)
            a = a + 1
        Loop While (a < 20)
    End Sub
End Class
```

To view IntelliTrace data, you must break execution of your application.

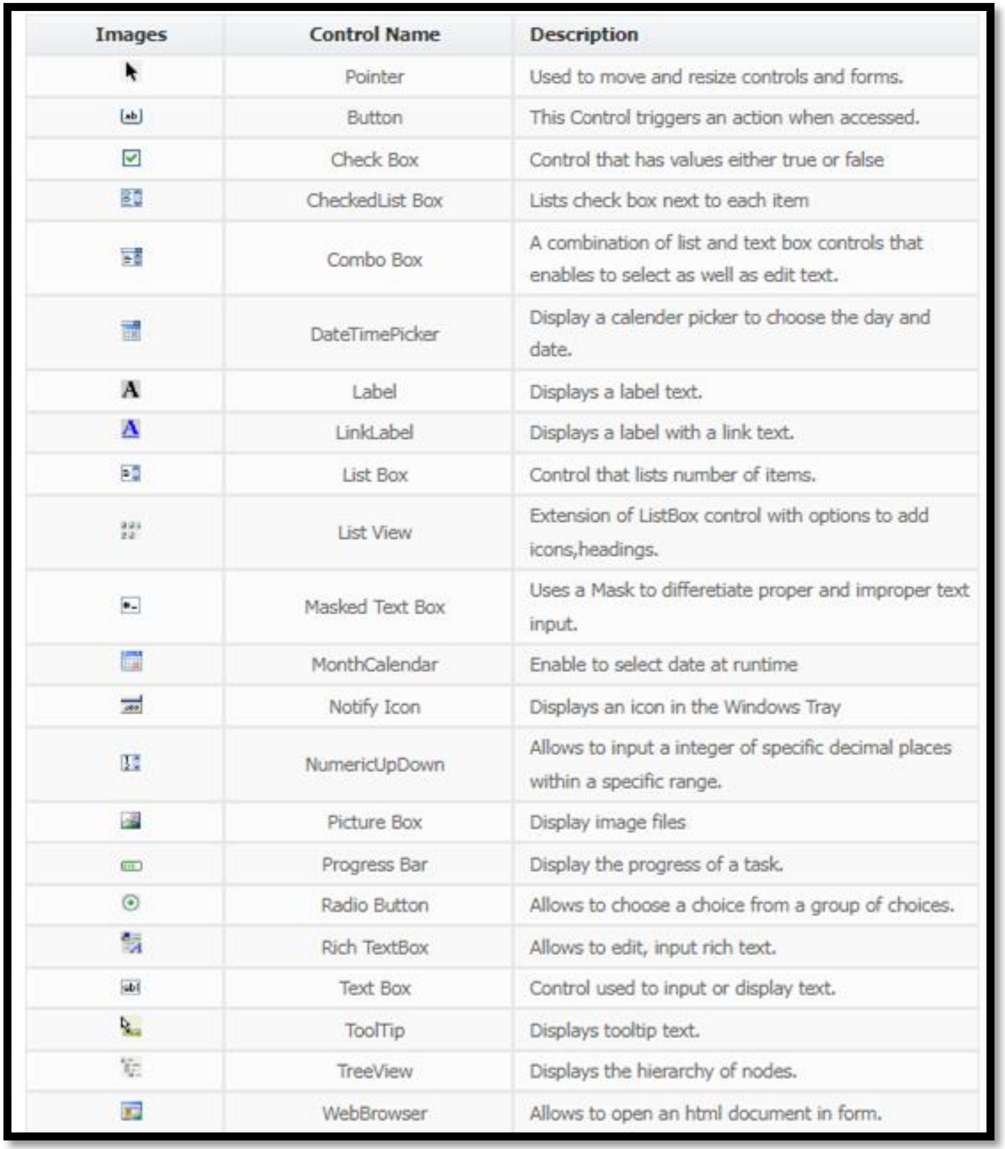
Design and Develop by Assistant Professor Ankit Rami Any Query Contact - 8460467193

Basic Tools Type in VB.Net

- Types of Tools Category



- All Windows Forms
- Common Controls
- Containers
- Menus & Toolbars
- Data
- Components
- Printing
- Dialogs
- Reporting
- WPF Interoperability
- Visual Basic PowerPacks
- General



Images	Control Name	Description
↖	Pointer	Used to move and resize controls and forms.
ab	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.
A	Label	Displays a label text.
A	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.
	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.
abl	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.

Reference Link

- https://www.tutorialspoint.com/vb.net/index.htm
- https://www.javatpoint.com/vb-net
- https://www.hscripts.com/tutorials/vbnet/tool-box.html
- https://www.hscripts.com/tutorials/vbnet/if-then.html
- https://www.hscripts.com/tutorials/vbnet/do-while-loop.html
- https://www.tutlane.com/tutorial/visual-basic