

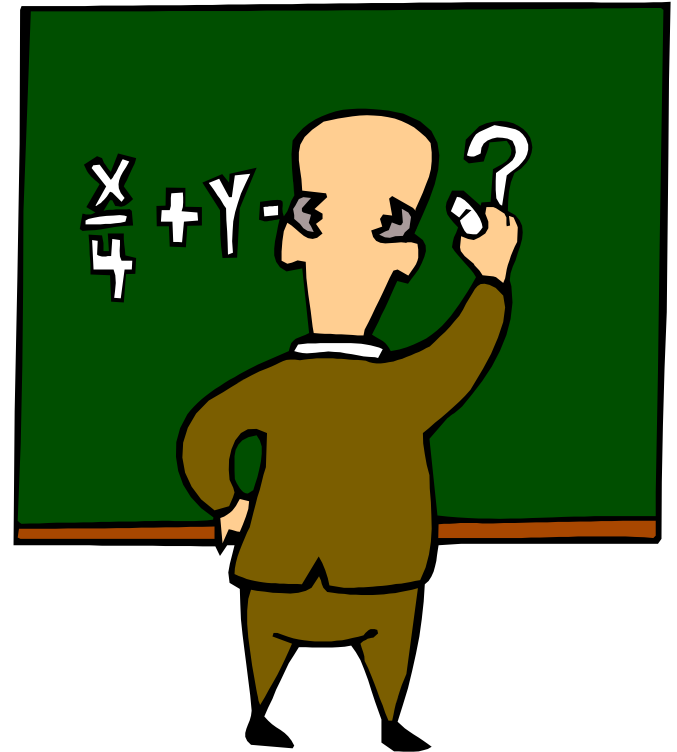
# Let's Get Visual!



Learn about variables... what they are and how to use them.

# What Are Variables?

- A variable is a word or symbol that represents a value to the computer.
- Just like in Logo, you can call a variable anything, and set its value to be a number or a string of one or more words.





# Declaring a Variable in VB

- To tell VB to make a variable “num” that holds a whole number (an integer), you say:

**Dim num As Integer**

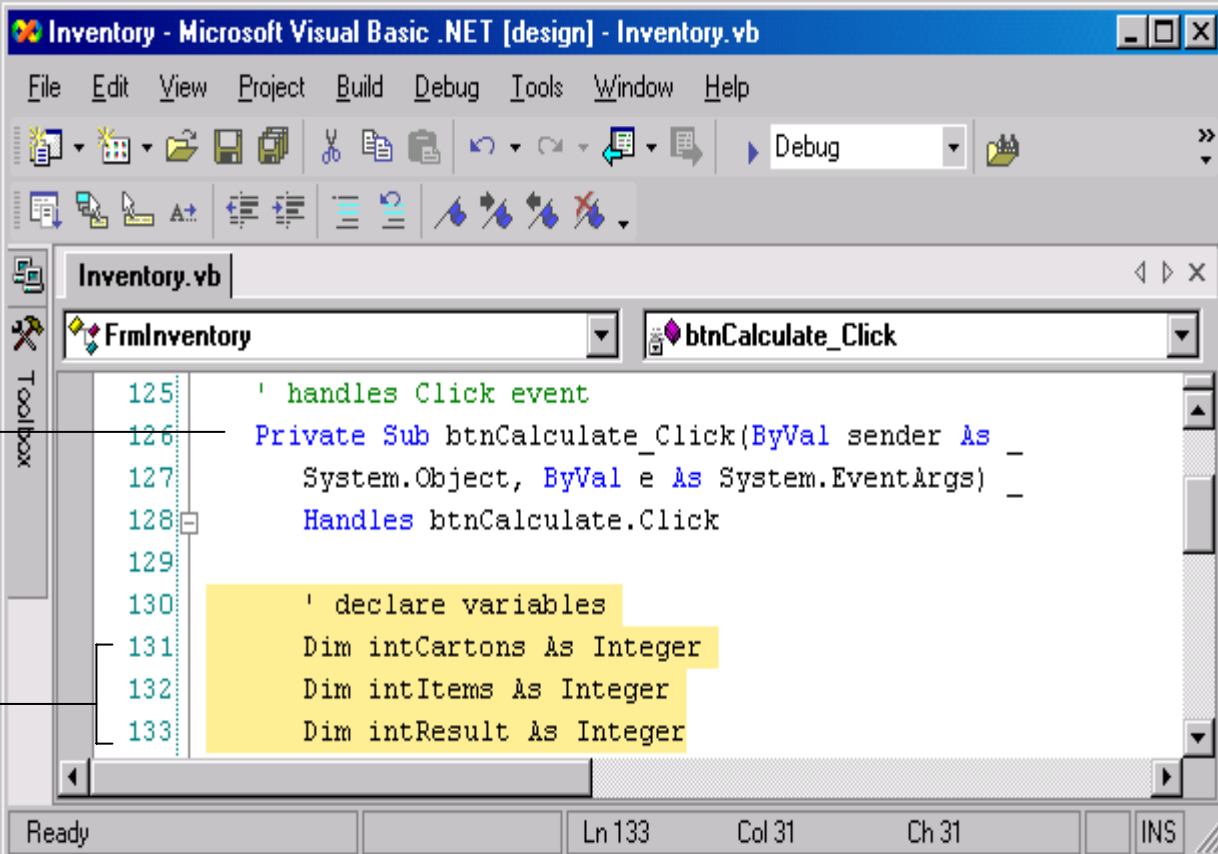
- To tell VB to make a variable “sentence” that holds a string of word(s) (a string), you say:

**Dim sentence As String**

- To tell VB to make a variable “deci” that holds a decimal number (a floating point number), you say:

**Dim deci As Float**

# For Example...



The screenshot shows the Microsoft Visual Basic .NET IDE with the following code in the `Inventory.vb` file:

```
125 ' handles Click event
126 Private Sub btnCalculate_Click(ByVal sender As _
127     System.Object, ByVal e As System.EventArgs) _
128     Handles btnCalculate.Click
129
130     ' declare variables
131     Dim intCartons As Integer
132     Dim intItems As Integer
133     Dim intResult As Integer
```

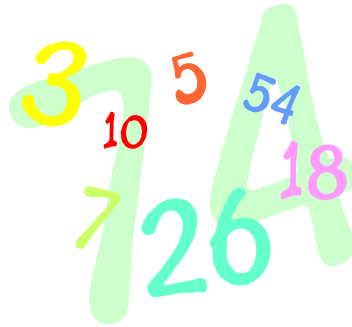
The code is annotated with two callouts:

- Click event handler:** A blue box pointing to the `Private Sub btnCalculate_Click` method signature.
- Variable declarations:** A blue box pointing to the `Dim` statements for `intCartons`, `intItems`, and `intResult`.

The status bar at the bottom indicates the current position is `Ln 133 Col 31 Ch 31`.

# Giving a Variable a Value

So we've declared three variables, but they don't have values yet. Here's a couple ways you can put values into variables:



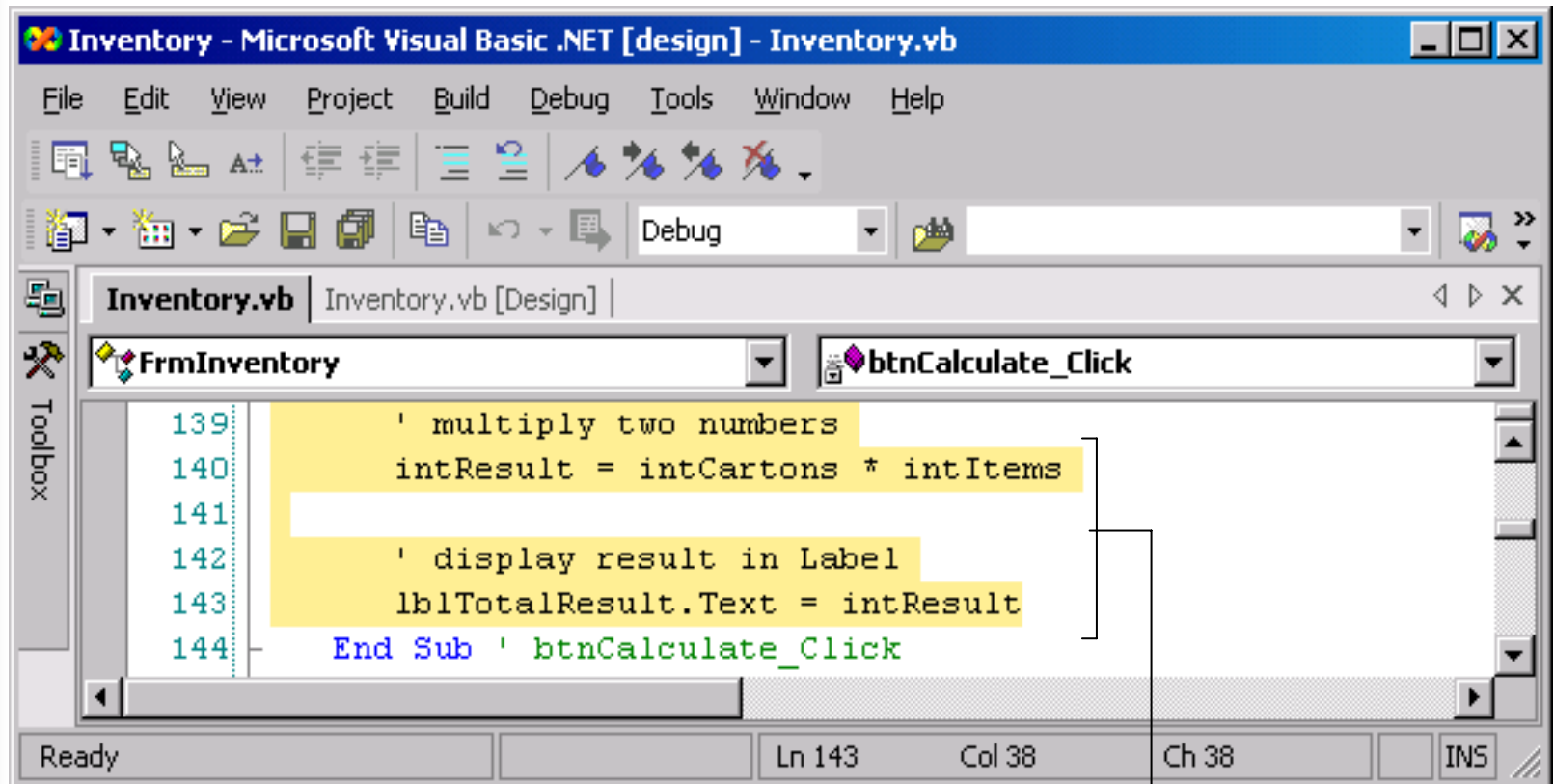
- Set the value of the variable to be the same as the contents of a text box.
- Set the value of the variable equal to any number.
- Set the value of the variable equal to the sum, product, difference, etc. of any other numbers or text box values.

# When Would You Need A Variable?

- When you need to store a value when going from one step of a program to another.
- When there's a value you want to use over and over.
- Can you think of anymore?



# Variable Example



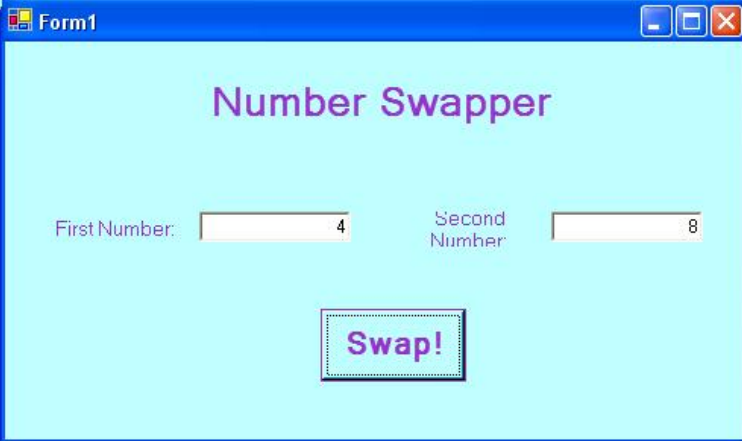
The screenshot shows the Microsoft Visual Basic .NET IDE. The title bar reads "Inventory - Microsoft Visual Basic .NET [design] - Inventory.vb". The menu bar includes File, Edit, View, Project, Build, Debug, Tools, Window, and Help. The toolbar contains various icons for file operations and development. The main window displays the code for "Inventory.vb" in the "Design" view. The code is as follows:

```
139      ' multiply two numbers
140      intResult = intCartons * intItems
141
142      ' display result in Label
143      lblTotalResult.Text = intResult
144  End Sub ' btnCalculate_Click
```

The status bar at the bottom indicates "Ready", "Ln 143", "Col 38", "Ch 38", and "INS".

Calculating  
and displaying  
the result

# Assignment: Number Swapper

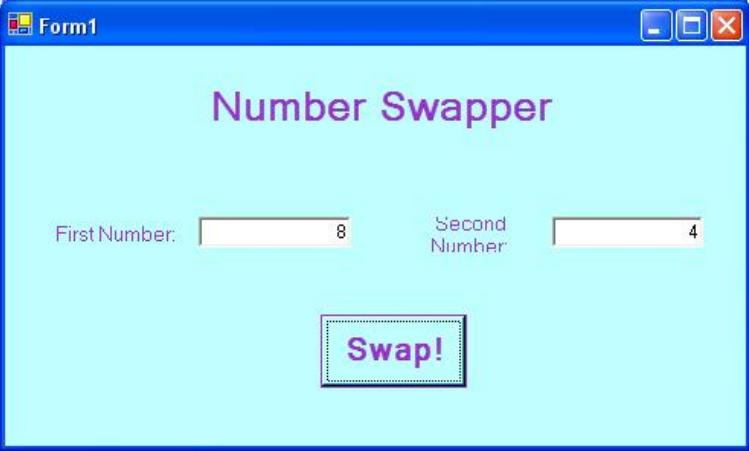


Form1

Number Swapper

First Number:  Second Number:

Swap!



Form1

Number Swapper

First Number:  Second Number:

Swap!

- Create the GUI at left.
- Double-click the button to open the code.
- Declare a variable to hold intermediate values.
- Use that variable to write the code to swap the values in the text boxes.

# Let's Get Visual!

A decorative horizontal bar consisting of a series of vertical rectangular segments in various colors including black, blue, light blue, teal, yellow, and dark blue, arranged in a slightly wavy pattern across the width of the slide.

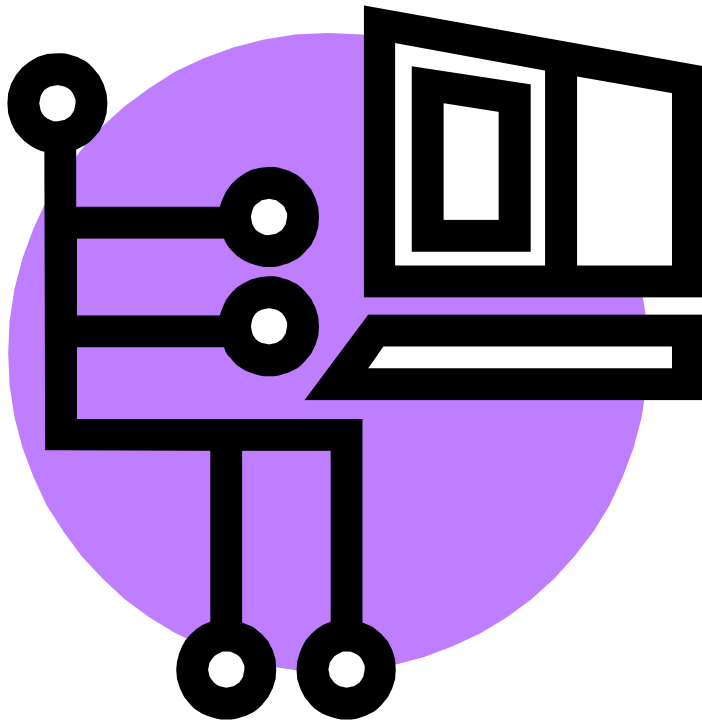
Learn what a conditional statement is and how to use it.

# Conditional Statements

- There are tons of real-life situations where you will do something in one scenario and do nothing otherwise.
- How many of these situations can you think of?



# Conditional Statements



- There are a million situations in programming when you'll want the computer to behave this way.
- Programming languages like VB have an if-then conditional statement built-in so that you can communicate the situation to the computer.

# Syntax: If Then Statement

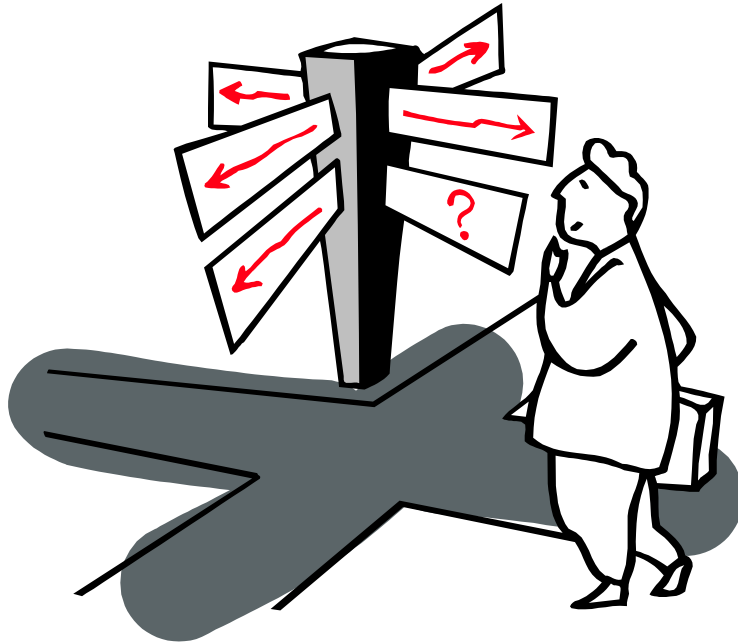
## Syntax

```
If condition Then  
    [ statements ]  
End If
```

Figure 7.5 If...Then statement syntax.

```
If intStudentGrade >= 90 Then  
    lblDisplay.Text = "A"  
  
End If
```

# Conditional Statements



- More often, you will perform one action in one scenario and a completely different action otherwise.
- For this situation, VB has an if-then-else statement to communicate the situation to the computer.

# Syntax: If Then Else Statement

## Syntax

```
If condition Then  
    [ statements ]  
Else  
    [ statements ]  
End If
```

Figure 7.7 If...Then Else statement syntax.

```
If intStudentGrade >= 90 Then  
    lblDisplay.Text = "A"  
Else  
    If intStudentGrade >= 80 Then  
        lblDisplay.Text = "B"  
    Else  
        If intStudentGrade >= 70 Then  
            lblDisplay.Text = "C"  
        Else  
            If intStudentGrade >= 60 Then  
                lblDisplay.Text = "D"  
            Else  
                lblDisplay.Text = "F"  
            End If  
        End If  
    End If  
End If
```

# Syntax: If Then ElseIf Statement

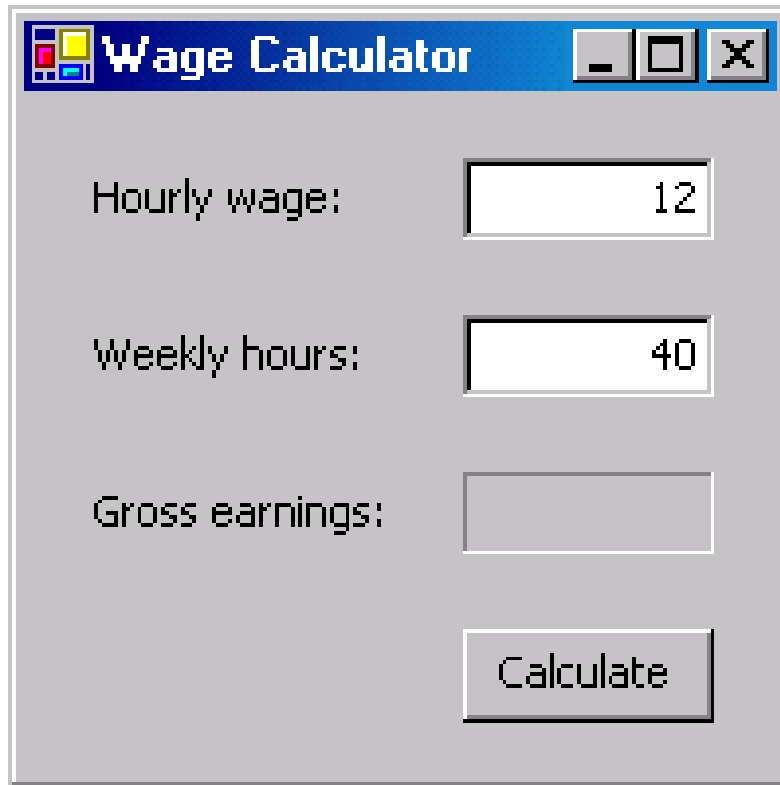
This structure is a fancy if-then-else statement that makes for much simpler code. It allows you to keep checking for true, but only until you find it, with very little code.

```
If intStudentGrade >= 90 Then
    lblDisplay.Text = "A"
ElseIf intStudentGrade >= 80 Then
    lblDisplay.Text = "B"
ElseIf intStudentGrade >= 70 Then
    lblDisplay.Text = "C"
ElseIf intStudentGrade >= 60 Then
    lblDisplay.Text = "D"
Else
    lblDisplay.Text = "F"
End If
```

# Relational Operators

<u>Conditional Operator</u>	<u>Code Example</u>	<u>Meaning</u>
$>$	<code>intX &gt; intY</code>	intX is greater than intY
$<$	<code>intX &lt; intY</code>	intX is less than intY
$\geq$	<code>intX \geq intY</code>	intX is greater than or equal to intY
$\leq$	<code>intX \leq intY</code>	intX is less than or equal to intY
$=$	<code>intX = intY</code>	intX is equal to intY
$\neq$	<code>intX \neq intY</code>	intX is not equal to intY

# Assignment: Wage Calculator



Hourly wage:

Weekly hours:

Gross earnings:

- Create the GUI at left.
- Double-click the button to open the code.
- Calculate the total wages earned.
- If the person worked more than 40 hours, s/he gets 1.5 times the wage per hour for that extra time.
- Use a conditional statement to check for this, and variables to store all the steps of the program.