<http://www.homeandlearn.co.uk/NET/nets1p14.html>

Now examine this:

**Dim number1 As Integer  
Dim number2 As Integer**

**number1 = 3  
number2 = 5**

That's code from Visual Basic Net. It's VB's way of setting up (or declaring) variables.

Here's a breakdown of the variable Declaration:

**Dim**

Short for Dimension. It's a type of variable. You declare (or "tell" Visual Basic) that you are setting up a variable with this word. We'll meet other types of variables later, but for now just remember to start your variable declarations with Dim.

**number1**

This is the cardboard box and the sticky label all in one. This is a variable. In other words, our storage area. After the Dim word, Visual Basic is looking for the name of your variable. You can call your variable almost anything you like, but there are a few reserved words that VB won't allow. It's good practice to give your variables a name appropriate to what is going in the variable.

**As Integer**

We're telling Visual Basic that the variable is going to be a number (integer). Well meet alternatives to Integer later.

**Number1 = 3**

The equals sign is not actually an equals sign. The = sign means assign a value of. In other words, here is where you put something in your variable. We're telling Visual Basic to assign a value of 3 to the variable called number1. Think back to the piece of paper going into the cardboard box. Well, this is the programming equivalent of writing a value on a piece of paper

**Adding a Button to a Form**

Instead of double clicking the Button tool in the toolbox to add the control to the form, we'll explore another way to do it.

With your Form displayed in the Visual Basic Design environment, do the following:

* Click on the Button tool in the toolbox with the left hand mouse button, but click only once
* Move your mouse to a blank area of your form - the mouse pointer will turn into a cross
* Press and hold down the left mouse button
* Drag across the form with the button held down
* Let go of the mouse button when you're happy with the size
* A Button is drawn

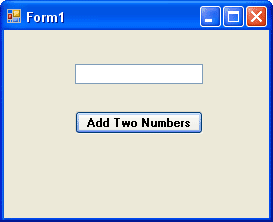
You can use the above method to draw most of the controls onto the form - labels, Buttons, textboxes, etc.

The Button control, just like all the other controls we've seen so far, has a list of properties. One of these properties is the Text property. At the moment, your button will say "Button 1". You can change that to anything you like.

* Click on the Button to highlight it
* Click on Text in the Property Box
* Click in the box next to the word "Text"
* Delete the word "Button 1"
* Type "Add two numbers"
* Click back on the Form

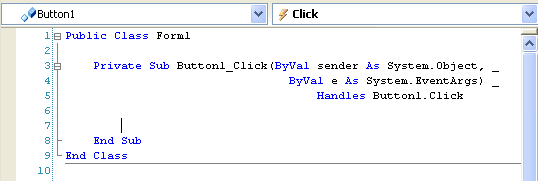
Now add a Textbox to your form using one of the methods outlined (either double-click, or draw).

Your Form should now look something like this:

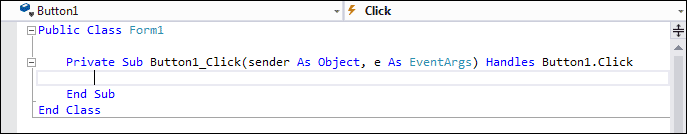


The Font property of the Button has also been changed, here, in exactly the same way as we changed the Font property of the Label and Textbox previously. The Text for the Textbox control has had its default Text (Textbox 1) deleted.

To get our first look at the code window, double click your Button control. The code window will appear, and will look like this in version 2010:



And this in version 2012/13:



In the 2010 version, notice that we've used the underscore character ( \_ ) to spread the code over more than one line. You can do this in your own code, if it becomes too long. But you don't have to.

In the 2012/13 version, notice that the word **ByVal** is missing between the round brackets. Don't worry about what this means, but we'll cover it much later. (It's still ByVal, in case you're curious, but it's missing because ByVal the default type.)

The part to concentrate on for the moment is where your cursor is flashing on and off. Because you double-clicked the Button control, the cursor will be flashing between the lines **Private Sub** … and **End Sub**.

Here's the part we're concentrating on:

**Private Sub Button1\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles Button1.Click**

**End Sub**

The part of the code we're interested in is highlighted in red in the code above. Notice, too, that the underscore character ( \_ ) has been used to spread the code over more than one line. You can do this in your own code, too, if it becomes to long:

**Private**

Private means that no other part of the programme can see this code except for our button

**Sub**

Short for Subroutine. The "Sub" word tells VB that some code follows, and that it needs to be executed

**Button1**

This is the name of our button. You might think that we've just erased the word "Button1" when we changed the Text property, so why does VB insist that it's still called Button1? We'll, the Name property of the control is the important one. If you change the Name property, VB will change this button name for you

**\_Click ( )**

This is something called an Event. In other words, when the button is clicked, the Click Event will fire, and the code we're going to write will be executed

**End Sub**

The subroutine ends right here. This signifies the end of our code

Click your mouse on the blank line after **Private Sub Button1\_Click**, etc, but before End Sub. Type the following code:

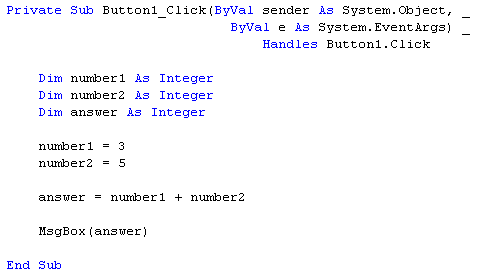
Dim number1 As Integer  
Dim number2 As Integer  
Dim answer As Integer

number1 = 3  
number2 = 5

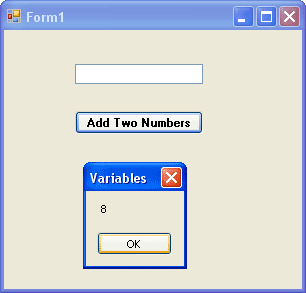
answer = number1 + number2

MsgBox answer

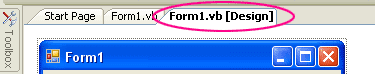
After typing all that, your code window should now look like this in version 2010 of Visual Studio Express. The only difference in version 2012 is the missing keyword **ByVal**:



Before we explore what's happening here, save your work and then click **Debug > Start** from the Visual Basic Menu, or press F5 on your keyboard. This will launch your programme. Click the Button once, and you should get the following:



Stop your programming, and return to the Design Environment. If you can't see your code, you can click the Tabs at the top of the Window, as in the image below:



Click the "Form1.vb [Design]" tab to see your Form.

OK, what happened there? Well, what happened is we've just wrote a programme to add two numbers together, and we displayed the result by using a Message Box - you very first real programme! But let's break that code down a little more.

* First, we started with the Dim word, indicating to Visual Basic that we wanted to set up a variable
* Then we gave the variable a name (number1)
* Next, we "told" VB that what is going inside the variable is a number (As Integer)
* Two more variable were set up in the same way, number2 and answer

After setting up the three variables, here's what we did:

* Told Visual Basic that what is going into the first variable was the number 3, and what is going into the second variable was the number 5. To put something into a variable, you use the equals ( = ) sign. But it's not really an equals sign - it's an assignment operator. You are assigning the value of 3 to the variable called number1

number1 = 3  
number2 = 5

The next part is a little bit more complicated, but not too complicated. What we wanted to do was to add two numbers together. So we said

number1 + number2

Visual Basic already knows how to add up: all we need to do is "tell" it to add up. We do the "telling" in the traditional, mathematical way - with the plus sign (+). What Visual Basic will do is to look at what we've stored inside number1, and look at what's inside number2. It's sees the 3, sees the five, and also sees the plus sign. Then Visual basic adds them up for you.

Except we also did something else. We said to Visual Basic "When you've finished adding up the two variables number1 and number2, store the result in that other variable we set up, which is called answer."

So, the whole line

answer = number1 + number2

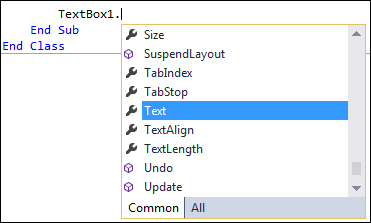
means: "Add the variable called number1 to the variable called number2. Then store the result in the variable called answer."

Think of it as working from the right-hand side of the equals sign first. Then when you have the answer, assign it the variable on the left of the equals sign.

The final part of the programme used Visual Basic's in-built Message Box. We'll learn a more about the Message Box later. For now, think of it as a handy way to display results.

Message boxes are quite handy when you want to display the result of some code. But we have a textbox on the form, and we might as well use that.

So delete the line: **MsgBox answer**. Type the word **Textbox1**, then type a full stop. You should see a drop-down box appear. This is a list of the Properties and Methods that the Textbox can use.



Scroll down until you see the word "Text". Double click the Text property and the drop-down box will disappear. (This drop-down box is known as IntelliSense, and is very handy. It means you can just select a property or method from the list without having to type anything.)

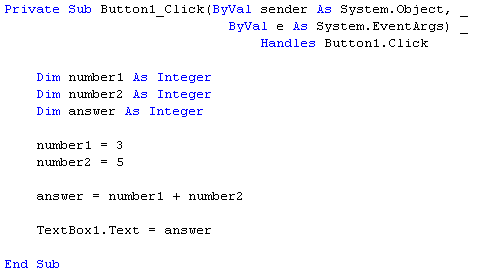
[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

The Text property you have chosen is the same Text property that you set from the Properties Window earlier. Here, we're setting the property with our code; before, we set it at design time. But the result is the same - the Text property of the textbox will be set to a value of our choosing.

To set a value, type an equals sign, then type a value for the Text property. We want the contents of the variable called **answer** to appear in the textbox. So the rest of the code is just this:

Textbox1.Text = answer

Your code window should then look like this:



Run your code again, and press the Button on the form. You should see the number 8 appear in the textbox.

OK, time for your first exercises. They're not too painful, and hopefully they'll giver you a better idea of what variables are. And besides, programming is about doing, not talking. So off we go!

### Exercise

Delete the values 3 and 5 and replace them with numbers of your own

### Exercise

Delete the plus sign in between number1 and number2, and replace them with each of the following in turn  
  
- (the minus sign)  
\* (the multiplication sign in VB is the asterisk sign)  
/ (the divide sign in VB is the forward slash)

### Exercise A

Set up another Integer variable. Give it the name number3. Assign a value of 10 to this new variable. Multiply the value of your new variable by the variable called answer. Display the result in your textbox.

[Answer to Exercise A](http://www.homeandlearn.co.uk/NET/vb_answers.html#ex1)

(Another way to assign values to variables is when you first set them up. You can do this:

Dim number3 As Integer = 10

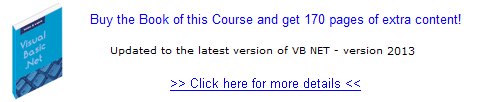
This is exactly the same as saying:

Dim number3 As Integer

number3 = 10

# String Variables

## An Introduction to Programming Strings

[](http://www.homeandlearn.co.uk/bookshop/net.html)

So we've learnt something about variables, what they are and how to set one up. We learnt about the word "integer", and that integer variables held numbers. But what if we don't want numbers? After all, our first Form asked users to type in their First Name and Last Name. Names are not numbers, so what do we do then? Well that's where Strings come in.

What is a String? Actually a string is nothing more than text. And if we want Visual Basic to store text we need to use the word "String". To set up a variable to hold text we need to use As String and not As Integer. If the information we want to store in our variables is a First Name and a Last Name, we can set up two variables like this.

Dim FirstName As String  
Dim LastName As String

Again we've started with the Dim word. Then we've called the first variable FirstName. Finally, we've ended the line by telling Visual Basic that we want to store text in the variable - As String.

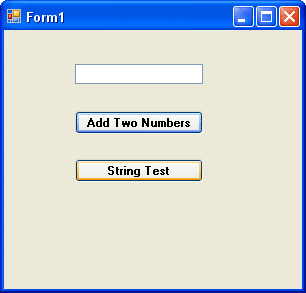
So we've set up the variables. But there is nothing stored in them yet. We store something in a variable with the equals sign ( = ). Let's store a first name and a last name in them

FirstName = "Bill"  
LastName = "Gates"

Here, we said to Visual Basic "Store the word 'Bill' into the variable FirstName and store the word 'Gates' into the variable called LastName. But pay attention to the quotation marks surrounding the two words. We didn't say Bill, we said "Bill". Visual Basic needs the two double quotation marks before it can identify your text, your String.

***So remember: if you're storing text in a variable, don't forget the quotation marks!***

To test all this out, add a new Button to your Form. Set the Text property of the Button to "String Test". Your Form would then look like this (it will look rather grey and dull in newer version of Visual Studio Express):



Double click your new button, and add the following code:

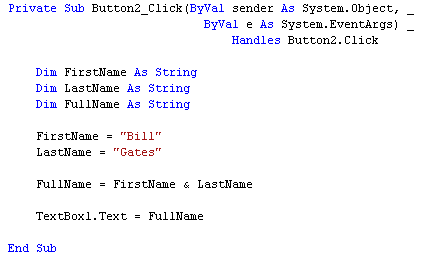
Dim FirstName As String  
Dim LastName As String  
Dim FullName As String

FirstName = "Bill"  
LastName = "Gates"

FullName = FirstName & LastName

Textbox1.Text = FullName

Your code window should now look like this (some of the first line has been cropped in the image below):



There's a line there that needs explaining

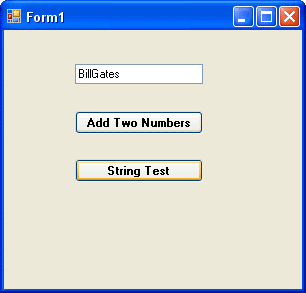
FullName = FirstName & LastName

In the two lines of code above that one, we stored the string "Bill" and the string "Gates" into two variables. What we're doing now is joining those two variables together. We do this with the ampersand symbol ( & ). The ampersand is used to join strings together. It's called Concatenation.

Once Visual Basic has joined the two strings together (or concatenated them), we're saying "store the result in the variable called FullName". After that, we tell VB to display the result in our Textbox.

So, once you've typed the code, start your programme and test it out.

Once the programme is running, Click the Button and see what happens. You should have a Form that looks something like this one:



The textbox displays the text stored in our variables, "Bill" and "Gates". We joined them together with the ampersand ( & ). But as you can see, the two words are actually joined as one. We can add a bit of space between the two words by using another ampersand. Change this line **FullName = FirstName & LastName** to this:

FullName = FirstName & " " & LastName

What we're saying here is join this lot together: the variable called FirstName and a single blank space and the variable called LastName. When you've finished concatenating it all, store the result in the variable FullName.

Notice that we don't surround FirstName and LastName with quotation marks. This is because these two are already string variables; we stored "Bill" into FirstName and "Gates" LastName. So VB already knows that they are text.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

### Exercise

Remove one of the ampersand symbols (&) from this line in your code:

FullName = FirstName & " " & LastName

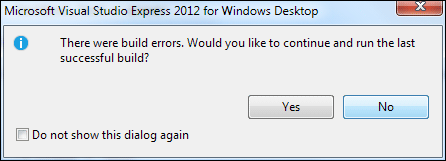
Move your cursor down a line or two. You should see that part of your code has a wiggly blue line under it:

Synatx error in VB NET

VB is telling you that it has problems with this line of code. If you hold your mouse over the wiggly blue line, VB tries to provide an explanation:

Error message

The explanations VB provides are sometimes enigmatic. But you will know that there is a problem. If you run the code, you'll get this popping up at you:



Click the NO button. Put the ampersand back in, and all will be well.

### Exercise B

Amend your code so that the textbox reads Gates Bill when the Command button is clicked.

[Answer to Exercise B](http://www.homeandlearn.co.uk/NET/vb_answers.html#ex2)

### Exercise C

Add another string variable to your code. The variable should hold a middle name. Display the first name, the middle name and the last name in the textbox.

[Answer to Exercise C](http://www.homeandlearn.co.uk/NET/vb_answers.html#exC)

Points to remember:

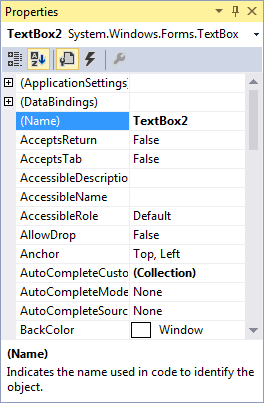
* Your variable names cannot include spaces. So the variable **MiddleName** would be all right, but **Middle Name** will get you an error message
* When you're putting text into your new variable, don't forget the two double quotes
* Remember to put in enough ampersands in your **FullName =** line of code

In the next part, we'll take a look at how to asign text from a textbox into our string variables.

# Assigning Textbox text to your Variables

Instead putting direct text into your variables, such as "Bill" or "Gates", you can get text from a textbox and put that straight into your variables. We'll see how that's done now. First, do this:

* Add a new textbox to your form
* With the textbox selected, locate the **Name** property in the Properties area:



The current value of the Name property is **Textbox2**. This is not terribly descriptive. Delete this name and enter **txtLastName**. Scroll down and locate the Text property. Delete the default text, and just leave it blank.

Click on your first textbox to select it. Change the Name property from **Textbox1** to **txtFirstName**.

What we've done is to give the two textboxes more descriptive names. This will help us to remember what is meant to go in them.

Unfortunately, if you view your code (click the Form1.vb tab at the top, or press F7 on your keyboard), you'll see that the blue wiggly lines have returned:

Syntax error in NET

If you hold your cursor of the Textbox1, you'll see this:

Error Message

It's displaying this message because you changed the name of your Textbox1. You now no longer have a textbox with this name. In the code above, change Textbox1 into **txtFirstName** and the wiggly lines will go away. (Change it in your Button1 code as well.) Your code should now read:

txtFirstName.Text = FullName

Run your programme again. If you see any error messages, stop the programme and look for the wiggly lines in your code.

We'll now change our code slightly, and make use of the second textbox. You'll see how to get at the text that a user enters.

Locate these two lines of code

FirstName = "Bill"  
LastName = "Gates"

Change them to this

FirstName = txtFirstName.Text  
LastName = txtLastName.Text

Remember: the equals ( = ) sign assigns things: Whatever is on the right of the equals sign gets assigned to whatever is on the left. What we're doing now is assigning the text from the textboxes directly into the two variables.

Amend your code slightly so that the Whole Name is now displayed in a message box. Your code should now be this:

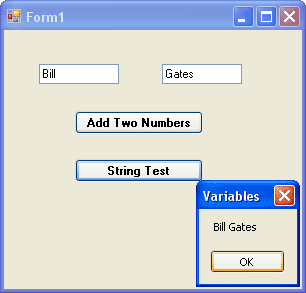
Dim FirstName As String  
Dim LastName As String  
Dim WholeName As String

FirstName = txtFirstName.Text  
LastName = txtLastName.Text

WholeName = FirstName & " " & LastName

MsgBox(WholeName)

Run your programme. Enter "Bill" in the first textbox, and "Gates" in the second textbox. Then click your "String Test" button. You should get this:



Before we changed the code, we were putting a person's name straight in to the variable FirstName

FirstName = "Bill"

But what we really want to do is get a person's name directly from the textbox. This will make life a whole lot easier for us. After all, not everybody is called Bill Gates! In the line **FirstName = txtFirstName.Text** that is what we're doing - getting the name directly from the textbox. What we're saying to Visual Basic is this

* Look for a Textbox that has the Name txtFirstName
* Locate the Text property of the Textbox that has the Name txtFirstName
* Read whatever this Text property is
* Put this Text property into the variable FirstName

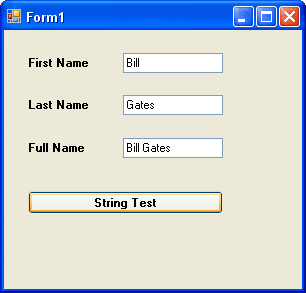
And that's all there is too reading values from a textbox - just access its Text property, and then pop it into a variable.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

### Exercise

* Add a third textbox to your form
* Change its Name property to txtWholeName
* Add labels to your form identifying each textbox (A quick way to add more labels is to use the toolbox to add one label. Then right click on that label. Choose Copy from the menu. Right click on the form, and select Paste.)
* Write code so that when the "String Test" button is clicked, the whole of the persons name is displayed in your new textbox

When you complete this exercise, your form should look like this one (we've deleted the first button and its code, but you don't have to):



# More about Variables in VB NET

We've met two variable types so far - **As String** and **As Integer**. But there are quite a few more you can use. Let's start by examining number variables.

Start a new project for this. If you have the old one displayed, you can click **File > Close Solution** from the menu bar. You will then be returned to the Start Page. Click the N**ew Project** button at the bottom. In the dialogue box, give your project a name.

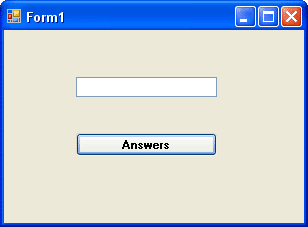
Put a textbox and a Button on your new form. Change the Properties of the Textbox to the following

**Name**: txtNumbers  
**Font**: MS Sans Serif, Bold, 10  
**Text**: just delete the default Textbox1, and leave the textbox blank

Change the Properties of the Button to the following:

**Text**: Answers  
**Font**: MS Sans Serif, Bold, 10

Click on the Form itself, and change it's **Text** property to "Testing Types". Your Form should look something like this:



Double click on the Button to bring up the code window. Type the following code for your Button (The **Button1\_Click** part is spread over three lines only for ease-of-reading on this web page. You can keep yours on one line in your code):

Private Sub Button1\_Click(ByVal sender AsSystem.Object, \_  
ByVale As System.EventArgs) \_  
Handles Button1.Click

Dim testNumber As Short

testNumber = Val( txtNumbers.Text )

MsgBox testNumber

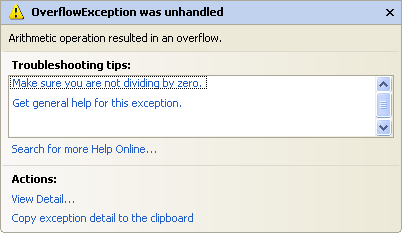
End Sub

Notice that there is a new Type of variable declared - As Short. This means "Short Integer". We'll see what it does. The **Val** part converts the Text into a number.

Run your programme. While it's running, do the following:

* Enter the number 1 into the textbox, and click the Answers button
* The number 1 should display in the Message Box
* Add the number 2 to the textbox and click the Button
* The number 12 should display in the Message Box
* Add the number 3 to the textbox and click the Button
* The number 123 should display in the Message Box
* Keeping adding numbers one at a time, then clicking the button

How many numbers did you get in the textbox before the following error message was displayed? (Click Break to get rid of it.)



You should have been able to enter **12345** quite safely. When you entered **123456** and clicked the button, that's when the error message displayed.

When you click the Break button, you are returned to the coding environment. You'll see the problem line highlighted in yellow:

Problem lines are highlighted in yellow

But your programme will still be running. So click **Debug > Stop Debugging** to return to the normal code window.

An Overflow happens when you try to put too much information into a variable that can't handle it.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

The reason we got an error message after just 6 numbers was because of the variable type. We had this:

Dim testNumber As Short

And it's As Short that is causing us the problems. If you use As Short you're only allowed numbers up to a certain value. The range for a Short variable is -32 768 to 32 767. When we entered 6 numbers, Visual Basic decided it didn't want to know. If you run your programme again, and then enter 32768, you'll get the same Overflow error message. If you change it once more to -32769, you'll get the error message as well. So it's not just 6 numbers a Short Type can't handle - it's 5 numbers above or below the values specified.

So what's the solution? Change the variable Type, of course!

Change the variable to this

Dim testNumber As Integer

Now start the programme and try again, adding numbers one at a time to the textbox, and then clicking the Command button. How far did you get this time?

If you started at 1 and added the numbers in sequence, you should have been allowed to enter 1234567890. One more number and Visual Basic gave you the Overflow error message, right? That's because variable types with As Integer also have a limitation. The range you can use with the As Integer variable type is -2,147,483,648 to 2,147,483,647. If you want a really, really big number you can use As Long.

Dim testNumber As Long

But these will get you whole numbers. Leave your number on As Integer. Run your programme again and enter a value of 123.45 into your textbox. Press the button and see what happens.

VB will chop off the point 45 bit at the end. If you want to work with floating point numbers (the .45 bit), there are three Types you can use:

Dim testNumber As Single  
Dim testNumber As Double  
Dim testNumber As Decimal

Single and Double mean Single-Precision and Double-Precision numbers. If you want to do scientific calculations, and you need to be really precise, then use Double rather than Single: it's more accurate.

The **As Decimal** Type is useful when you want a precise number of decimal places. It's not as accurate as the Double Type, though.

In terms of the space used in the computer's memory, Short Types use 2 Bytes, Integer Types use 4 Bytes, Long Types use 8 Bytes, Single Types use 4 Bytes, Double Types use 8 Bytes, and Decimal Types use 16 Bytes.

### Exercise

Write a programme to calculate the following sum.

0.123345678 \* 1234

Use the Single Type first, then change it to **As Double**. Use a Message box to display the answer. Was the number rounded up or rounded down for the Single Type?

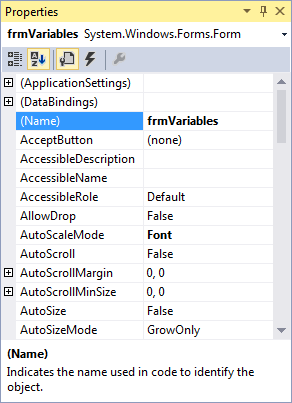
In the next part, we'll get some more practise with variables.

# Using Variables in your VB NET Code

In this next section, we're going to learn how to transfer the contents of one textbox to another textbox. We'll also learn to transfer the text from a label to a textbox, and whatever was in the textbox we'll transfer it to a label. This will get us a little more practise with variables, and how to use them.

Ok, start a new Visual basic project. You should know how to do this by now, and what the design environment looks like. But you should have a plain grey Form on your screen. By default it will be called Form1.

Make sure the Form is selected (has it got the white squares around it?), and the click the **Name** property in the Properties window. Change the Name of the form to **frmVariables**.



Set the Text property of the Form to "Transferring information". You can choose any background colour you like for the form, or leave it on the default.

Put the following controls on the Form, and change their properties to the one's specified below (NOTE: **lbl** is short for label):

**Textbox**

**Name**: txtVariables  
**Font**: MS Sans Serif, Bold, 10  
**Text** Delete the default text "Text1" and leave it blank

**Label**

**Name**: lblTransfer   
**BackColor**: A colour of your choice  
**Text**: Label Caption  
**Font**: MS Sans Serif, Bold, 10

**Button**

**Name**: btnTransfer  
**Text**: Transfer

The height of your controls is entirely up to you.

If you double click your Button to bring up the code window, you will see that the first line of the code no longer says Button1\_Click (etc). The first line should say this

Private Sub **btnTransfer\_Click** (ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles btnTransfer.Click

End Sub

The reason it has changed is because you changed the Name property of the Button. The button now has the Name **btnTransfer**. If you wanted to, you could change the Name property back to Button1. Then when you double clicked the button, the code window would pop up and the first line would be Button1\_Click(etc ).

What we're going to do now is to transfer the Text on the label ("Label Caption") to our empty textbox. And all with the click of a button.

As you'll see, there isn't much code.

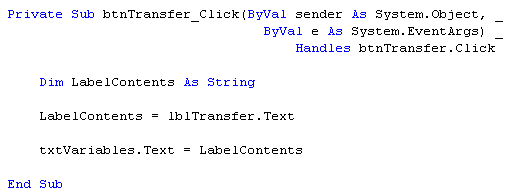
Type the following into your code window:

Dim LabelContents As String

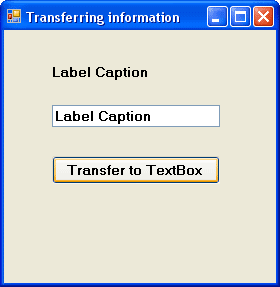
LabelContents = lblTransfer.Text

txtVariables.Text = LabelContents

Your code window should now look something like this:



Now Run your programme and test it out. When you click on the "Transfer" button, you should see that the contents of the label will be inserted into the textbox:



But let's break the code down and see what's going on.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

**Dim LabelContents As String**

Here is where we set up a variable called LabelContents. Because it will be holding text, we've used the variable type As String.

**LabelContents = lblTransfer.Text**

Here is where we put something into our empty variable. We changed the Name property of our Label from the default Label1 to lblTransfer. A Label has lots of properties you can manipulate. One of those properties is the Text property. After you typed the word "lblTransfer" and then typed a full stop, you probably saw a drop down box appear. Inside the box is a list of all the properties and Methods that a Label has. We wanted to manipulate the Text property of our label so we selected the word Text after the full stop. So we were saying "Access the value of the Text property of the label called lblTransfer, and put this value into the variable called LabelContents." Because our Text was ""Label Caption", the variable LabelContents now holds the text "Label Caption."

**txtVariables.Text = LabelContents**

Finally, we want to transfer whatever is in the variable LabelContents to the Textbox. Our Textbox is called txtVariables. Again, after typing the full stop the drop down box would appear, showing you a list of all the properties that a Textbox has. The one we're interested in is the Text Property. So we're saying, "Take whatever text is in the variable LabelContents, and transfer it to the Text property of the Textbox called txtVariables.

And with three lines of code we can transfer text from a label to a Textbox. But can we do it the other way round? Can we transfer whatever is in a Textbox to a Label? Well, sure we can.

Add another button to your form. Change its Name property from Button1 to cmdTransferToLabel, and change the Caption property to "Transfer To Label". Again, there's just three lines of code.

So double click your new button to bring up the code window. Then type in the following code:

Dim TextBoxContents As String

TextBoxContents = txtVariables.Text

lblTransfer.Text = TextBoxContents

Now, see if you can work out how it works. It's the same thing as the first three lines of code: set up a variable, transfer the Text property of the Textbox to the variable, transfer the variable to the Text property of the Label. Run your programme and test it out. Type something into the Textbox, and then click the "Transfer To Label" button.

### Exercise E

A button also has a Text Property. Write code to transfer the Text property of a button to the Textbox. It's probably better for this exercise to create a new Button. Set its Name property to whatever you like. And give its Text Property a new value (The Text property will be Button1 by default) .

But the process is exactly the same as the two bits of code above - you should only need 3 lines of code for this exercise.

* Set up a Variable
* Transfer the Text property of the button to the variable
* Transfer the variable to the Textbox

[Answer to Exercise E](http://www.homeandlearn.co.uk/NET/vb_answers.html#exE)

In the next part, we'll start a Calculator project. This will get you some more practical experience of working with variables.

# A Calculator Project in VB NET

In the next few pages, you're going to create a Calculator. It won't be a very sophisticated calculator, and the only thing it can do is add up. What the project will give you is more confidence in using variables, and shifting values from one control to another. So create a new project, call it **Calculator**, and let's get started.

### Designing the Form

Let's design the form first. What does a calculator need? Well numbers, for one. A display area for the result. A plus sign button, an equals sign button, and a clear the display button.

Here's how our calculator is going to work. We'll have 10 button for the numbers 0 to 9. When a button is clicked its value will be transferred to a display area, which will be a Textbox. Once a number is transferred to the Textbox we can click on the Plus button. Then we need to click back on another number. To get the answer, we'll click on the equals sign. To clear the display, we'll have a Clear button.

If you haven't already, create a new project. Save it as Calculator. To your new form, first add ten Buttons (You can add one, then copy and paste the rest). The Buttons should have the following Properties:

**Name**: ***btn*** Plus a Number (btnOne, btnTwo, btnThree, etc)

**Text**: A number from 0 to 9. A different one for each button, obviously

**Font**: MS Sans Serif, Bold, 14

Next, add a Textbox. Set the following properties for the Textbox:

**Textbox**  
**Name:** txtDisplay  
**Font**: MS Sans Serif, Bold, 14  
**Text**: Erase the default, Textbox1, and leave it blank

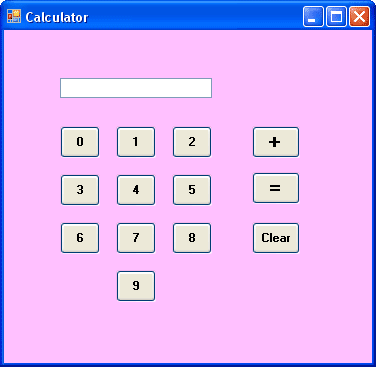
Three more buttons need to be added

**Plus Button**   
**Name**: btnPlus  
**Font**: MS Sans Serif, Bold, 14  
**Text**: +

**Equals Button**   
**Name**: btnEquals  
**Font** MS Sans Serif, Bold, 14  
**Text**: =

**Clear Button**  
**Name**: btnClear  
**Font** MS Sans Serif, Bold, 14  
**Text**: Clear

When your form design is finished, it might look something like this:



So if you wanted to add 5 + 9, you would click first on the 5. A 5 would appear in the textbox. Then you would click the + symbol. The 5 would disappear from the textbox. Next, click on the 9. The number 9 would appear in the textbox. Finally, click on the = symbol. The 9 would disappear from the textbox, and the answer to our sum would replace it. We would then click the Clear button to clear the display.

In the next section, we'll make a start on the VB NET code for the all those buttons.

# The Code for the VB NET Calculator

In the [previous part](http://www.homeandlearn.co.uk/NET/nets1p17.html), you designed the NET form for your calculator. We'll now make a start on the code.

You might be thinking that all this is terribly complicated at such an early stage. But it isn't really. All we are doing is transferring the Text Properties from the Buttons to the textbox. And you already know how to do that. The number buttons don't do anything else. All the work is done with the Plus button and the Equals buttons. And there's only two lines of code needed for the Plus button, and three for the Equals button.

For this to work, though, a little word about Scope in VB NET.

So far, when you've set up a variable, you've set them up behind a Private Subroutine. Like this:

Private Sub Button1\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles Button1.Click

Dim MyVariable As String

End Sub

Suppose you had another button on the form, Button2, and the code was this

Private Sub Button2\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles Button2.Click

Dim MyOtherVariable As String

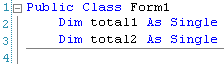
End Sub

How can you access what's in **MyVariable** from Button2? The answer is, you can't. It's like two people sitting at desks in cubicles. Each person has written something on a piece of paper. They can't see into the other person's cubicle, only whatever is their own cubicle. So how do they share their information?

Well suppose there is a screen in front of them. A big screen. They can both see the screen in front of them; it's each other they can't see. What they could do is project their information onto the screen. Then one person could see what the other has written.

Similarly, in VB you can set up your variable declarations outside of the code for a Button. That way, more than one Button can see the code.

You can place your variable declarations right at the top of the code window, just beneath the line that begins "Public Class Form1". We'll set up two Integer variables there, **total1** and **total2**:



Now all the controls on your form can see these two variables. Those Buttons you set up can put something in them, and every button has the ability to see what's inside them.

(Incidentally, if you want to know how to add the line numbers from the image above, click on **Tools > Options** from the menu at the top of Visual Studio Express. From the dialogue box that appears, click on **Text Editor** on the left. Then click on **All Languages**. From the area on the right, put a check in the box for **Line Numbers**.)

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

### The 0 to 9 Buttons

The Buttons with the Text 0 to 9 only need to do one thing when the button is clicked - have their Text Properties transferred to the Textbox.

So double click the 0 Button and enter the following code:

Private Sub btnZero\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles btnZero.Click

txtDisplay.Text = btnZero.Text

End Sub

This code will transfer the Text Property of a Button called btnZero to the Text Property of a Textbox called txtDisplay.

Run your programme and try it out. When the programme is running, click the 0 button to see that it does indeed transfer the Text on the Button to the textbox

Except, there's a problem with that code. If you wrote similar code for all ten of your number buttons, the calculator wouldn't be right. Why is that? Have you spotted what's wrong? It's a good idea to set this book aside for a while and think about why this code on it's own wouldn't work. In fact you could write code for a few more of the number buttons and test it out.

What happens when you transfer the number 2 to the Textbox, and then click the number 3? The number 2 will disappear, to be replaced by the number 3. Which is all right if all you wanted to do was add up single numbers, but not much good if you actually wanted the number 23 in the Textbox. With this code, you could have either the number 2 in the Textbox or the number 3, but not both!

So how do we solve this problem? How do we fix it so that we can have two or more numbers appearing in our Textbox?

What we need is a way to get whatever is in the Textbox to stay where it is, and not disappear on us when we click a different number. It's quite easy. It's this:

txtDisplay.Text = txtDisplay.Text & btnZero.Text

So now we're saying the textbox doesn't just contain the Text on the Button. It must also keep whatever is inside the textbox as well.

So what you need to do now is to add that code to all of your ten number Buttons. Obviously it won't be exactly the same. For the button called btnOne the code would be this:

txtDisplay.Text = txtDisplay.Text & btnOne.Text

When you've finished coding all ten buttons, run the programme and click all ten number button to see if they do indeed transfer the numbers on the caption to the textbox. Not only that, but test to see if you can have more than one number in the textbox.

Now that we can get numbers into our Textbox display area, we'll write code to do something with those numbers - add them together, in other words. We'll do that in the next part. Click the link below to move on.

# Coding the Plus Button

Let's remind ourselves how our calculator from the [previous section](http://www.homeandlearn.co.uk/NET/nets1p18.html) works works. To add up 5 + 9, we'd do this:

1. Click first on the 5
2. A 5 appear in the textbox
3. Click the + symbol
4. The 5 disappears from the textbox
5. Click on the number 9
6. A 9 appears in the textbox
7. Click on the = symbol
8. The 9 disappears from the textbox
9. The answer to 5 + 9 appears in the textbox
10. Click the "Clear" button to clear the textbox

We've done numbers 1 and 2 on that list. We're now going to do numbers 3 and 4 on the list. What we're trying to do is this: Click on the Plus symbol and make the number in the Textbox disappear. Before the number vanishes, we can store it in a variable. The variable we're going to be storing the number in is one of those variable we set up at the top of the code. It's this one:

Dim total1 As Integer

We've already seen how to retain a value from a textbox and add it to something else:

txtDisplay.Text = txtDisplay.Text & btnZero.Text

Here, we kept the value that was already in the textbox and joined it to the Text property of the button.

We can do something similar if we want to retain a value that is already in a variable. Examine this:

variable1 = variable1 + 1

The "**= variable1 + 1**" part just says "Remember what is in the variable variable1, and then add 1 to it. So if variable1 contain the number 3, what would variable1 now hold after that bit of code is executed? The whole code might be this:

variable1 = 3  
variable1 = variable1 + 1

(If you don't know the answer to that, please send an email and ask for some further clarification on the subject.)

The above is known in programming terms as "Incrementing a variable". There is even a shorthand you can use:

variable1 += 1

This says "variable1 equals variable1 plus 1". Or "Add 1 to variable1". You can also use the other mathematical operators with the same shorthand notation:

variable1 = 10  
variable1 \*= 3

This new code says "Multiply whatever is inside of variable1 by 3".

The shorthand notation can be tricky to read (and to get used to), so we won't use it much. But it's there is you want it.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

Back to our code.

If we're going to be adding two numbers together, we need Visual Basic to remember the first number. Because when we click the equals sign, that first number will be added to the second number. So if we haven't asked VB to remember the first number, it won't be able to add up.

The variable we're going to be storing the first number in is total1. We could just say this:

total1 = txtDisplay.Text

Then whatever is in the textbox will get stored in the variable total1.

Except we want VB to remember our numbers. Because we might want to add three or more numbers together: 1 + 2 + 3 + 4. If we don't keep a running total, there's no way for our programme to remember what has gone before, it will just erase whatever is in total1 and then start again.

So just like the code above (varaible1 = variable1 + 1), we need to "tell" our programme to remember what was in the variable. We do it like this:

total1 = total1 + Val(txtDisplay.Text)

That Val( ) part just makes sure that a number in a textbox is kept as a number, and not as text. It's short for Value. The important part is the total1 + txtDisplay.Text. We're saying "The variable total1 contains whatever is in total1 added to the number in the textbox." An example might clear things up. Suppose we were adding 5 + 9. Now, suppose total1 already contained the number 5, and that number 9 was in the textbox. It would work like this:

Finally, we need to erase the number in the textbox. To erase something from a textbox, just set its Text property to a blank string. We do this with two double quotation marks. Like this:

txtDisplay.Text = ""

That tiny bit of code will erase the Text a textbox. Another way to erase text from a textbox is to use the Clear method. After you typed a full stop, you probably saw the drop down list of Properties and Methods. Scroll up to the top, and locate the word Clear. Double click "Clear" and the drop down list will close. Hit the return key and VB adds two round brackets to your code:

txtDisplay.Clear( )

Notice that we're not setting the textbox to equal anything. We're using something called a Method. You can tell it's a Method because there's a purple block icon next to the word. A Method is a built-in bit of code that VB knows how to execute. In other words, it knows how to clear text from a textbox. You'll learn more about Methods later.

So the whole code for our Button called btnPlus is this:

Private Sub btnPlus\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles btnPlus.Click

total1 = total1 + Val(txtDisplay.Text)

txtDisplay.Clear()

End Sub

Add that code to your Plus button. All we've done with that code is to store numbers into our variable total1 and then erase whatever was in the textbox.

We now need to add the numbers up.

### Exercise F

Write the code for the equals button. There's only three lines in total, and here's a little help.

You need to use the other variable that was set up at the top of the coding window, total2. The variable total1 will be added to whatever is total2

The first line of code will be this

total2 = total1 + (something missing here)

Your job is to supply the missing code. In other words, replace "(something missing here)"

Remember that **total1** contains the first number to be added. And you know where that came from. The only thing left to do is to add the second number.

For the second line of code, you need to transfer the **total2** variable to the textbox.

For the third line of code, all you are doing is resetting the variable total1 to zero. That's because after the equals button has been pressed, we have finished adding up. If you wanted to do some more adding up, that total1 will still hold the value from the last time. If you reset it to zero, you can start adding some new numbers.

The only thing left to do is to write a single line of code for the Clear button. All you are doing there is erasing text from a textbox. Which you already know how to do.

[Answer to Exercise F](http://www.homeandlearn.co.uk/NET/vb_answers.html#exF)

When you're finished, you should have a simple calculator that can add up integers. In the next section, we'll take a look at Conditional Logic, and why it's so important for your programming skills

**Conditional Logic - If Statements**

What is conditional logic? Well, it's something you use in your daily life all the time, without realising you're doing it. Suppose that there is a really delicious cream cake in front of you, just begging to be eaten. But you are on a diet. The cake is clearly asking for it. So what do you do, eat the cake and ruin your diet? Or stick to your diet and let somebody else have that delicious treat? You might even be saying this to yourself:

**If**I eat the cake **Then** my diet will be ruined

**If** I don't eat the cake **Then** I will be on course for a slimmer figure

Note the words **If** and **Then** in the above sentences. You're using conditional logic with those two words: "I will eat the cake on condition that my diet is ruined". Conditional logic is all about that little **If** word. You can even add Else to it.

**If** I eat the cake **Then** my diet will be ruined

**Else**

**If** I don't eat the cake **Then** I will be on course for a slimmer figure

And that is what conditional Logic is all about - saying what happens if one condition is met, and what happens if the condition is not met. Visual Basic uses those same words - If, Then, Else for conditional Logic. Let's try it out.

Start a new project. Give it any name you like. In the design environment, add a Button to the new form. Double click the button and add the following code to it:

**Private Sub Button1\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles Button1.Click**

**Dim firstname As String**

**firstname = "Bill"  
If firstname = "Bill" Then MsgBox("firstname is Bill")**

**End Sub**

Run the programme and see what happens. You should get a Message Box with the words "firstname is Bill" in it.

What we did was to set up a string variable and put the name "Bill" into it. When then used conditional logic to test what was in the variable. In fact, we used an If statement. We said:

If the variable called firstname holds the value "Bill" Then display a Message Box

We can tidy that up a bit, because a single line of code can get very long with If statements. We can use this format instead.

**If firstname = "Bill" Then**

**MsgBox "firstname is Bill"**

**End If**

That's a lot tidier. Note that we start a new line after the word Then.

1. The first line contains our condition: "If the following condition is met".
2. The second line is what we want to do if the condition is indeed met.
3. The third line tells Visual Basic that the If statement ends right here.

Try this. Delete the two quotation marks around the word Bill in the If Statement. Your code should now be this:

**Dim firstname as String**

**firstname = "Bill"**

**If firstname = Bill Then**

**MsgBox "firstname is Bill"**

**End If**

VB.NET puts a blue wiggly line under **Bill**. If you try to start your programme, you'll get a message box telling you that there were Build Errors, and asking if you want to continue.

Say No to return to the design environment. The reason for the blue wiggly line is that VB insists on you using double quotes to surround you text. No double quotes and VB insists it's not a string of text.

Change you code to this.

**firstname = "Phil"**

**If firstname = "Bill" Then**

**MsgBox "firstname is Bill"**

**Else**

**MsgBox "firstname is not Bill"**

**End If**

Now run the programme and see what happens when you click the button.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

You should have gotten a Message Box popping up saying "firstname is not Bill". The reason is that we included the Else word. We're now saying, "If the condition is met Then display one Message Box. If the condition is not met, display a different Message Box." Notice that the Else word is on a line of it's own.

Now, after you have tested your programme, try this. Add a textbox to your form. Then change this line in your code:

**firstname = "Phil"**

To this:

**firstname = Textbox1.Text**

What the code does is to transfer the text in the Textbox directly to the **firstname** variable. We can then test what is in the variable with an If statement.

When you've finished the code, test it out by typing the word "Bill" (with a capital B) into the textbox, and then clicking the button. Then try it with a lower case "b".

So far, we've explored only simple If statements, and we're going to leave it that way for now. But they can get quite complex, because you can have one If statement inside another, and multiple Else statements.

The code you have just wrote, however, does demonstrate how you can find out what is in a variable, and take action if the condition is either met or not met. We're now going to explore another way to do that - the Select Case statement.

# Select Case Statements

The Select Case statement is another way to test what is inside of a variable. You can use it when you know there is only a limited number of things that could be in the variable. For example, suppose we add another choice for that cream cake. We've only said that the consequences of eating the cream cake is that the Diet will be either "Ruined" or "Not Ruined". But what if we add another option - "Diet Not Tested". In other words, we ate the cake but refused to climb onto the scales to weigh ourselves!

With three choices, we can still use an If ... Else statement. But let's change it to a Select Case statement. Remember: all we are doing is testing what is inside a variable, in this case a variable called **creamcake**. Once we decide what is inside the variable, we can take some action. So let's look at how the Select Case works.

Create a Form with a button and a Textbox on it (If you have your form open from the [previous section](http://www.homeandlearn.co.uk/NET/nets1p20.html), then you can use this one). Double click the new button. You should see something like this appear.

Private Sub Button1\_Click(ByVal sender As System.Object, \_  
ByVal e As System.EventArgs) \_  
Handles Button1.Click

End Sub

Between the button Sub and End Sub code add the folowing

Dim creamcake As String  
Dim DietState As String

creamcake = TextBox1.Text

Select Case creamcake

Case "Eaten"  
DietState = "Diet Ruined"  
Case "Not Eaten"  
DietState = "Diet Not Ruined"  
Case Else  
DietState = "Didn't check"

End Select

MsgBox DietState

Run your code to test it out. Click inside your textbox and enter the word "Eaten". Then click your button to see what happens. Now enter the words "Not Eaten" and click your button. Next, try the word "eaten", with a lowercase "e".

So, how does the Select case work?

In the code above, we first set up two variables called **creamcake** and DietState. Next, we transfer whatever is in **Textbox1** to the variable **creamcake**. The Select Case begins on the next line:

Select Case creamcake

We tell Visual Basic that we want to start a Select Case statement by simply using the words "Select Case". This is enough to set up the statement. The variable **creamcake** follows the words Select Case. We're saying, "Set up a Select Case statement to test what is inside the variable called creamcake". The next line is this:

Case "Eaten"

We ask Visual Basic to check if the variable creamcake contains the word "Eaten". (Is it the Case that ... ?)

**If it is the Case that** creamcake contains the word "Eaten", then VB will drop down to the line or lines of code below and read that. If the variable creamcake doesn't contain the word "Eaten", the programme will skip the line or lines of code below and jump to the next Case.

The programme will continue to check all the words after Case to see if one of them contains what is in the variable creamcake. If it finds one, it will read the code below the Case word; if it doesn't find any matches, it will not do anything at all. In our code, we're checking these three Cases:

**Case** "Eaten"  
**Case**"Not Eaten"  
**Case** Else

Note also that it will only look for an exact match - "Eaten", but not "eaten".

The next line to examine is this:

Case Else

You can use the **Else** word after Case. If the programme hasn't found any matches, it will then execute the code below the Case Else line.

The final line to examing is this:

End Select

All we're doing here is to tell Visual basic to end the Select Case statement.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

So the Select Case checks a variable for any number of different choices. If a match is found, it will then execute code below the Case option it has found. In our code, we've just told the programme to store some text in the DietState variable. After the Select Case statement has ended we displayed the variable in a Message Box.

You can use Select Case statement with numbers, as well, and the **To** word comes in very handy here. If you were checking a variable to see if the number that was in the variable fell within a certain age-range, you could use something like this:

Select Case agerange

Case 16 To 21  
MsgBox “Still Young”  
Case 50 To 64  
MsgBox “Start Lying”

End Select

Here, a variable called agerange is being tested. It's being checked to see if it falls between certain values. If it does, then a message box is displayed.

### Exercise G

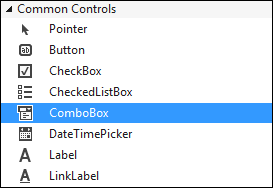
Add a new button to your form. Write a programme that tests if a person is a) A teenager, b) in their twenties, c) in their thirties, or d) none of the above.

[Answer to Exercise G](http://www.homeandlearn.co.uk/NET/vb_answers.html#exG)

A popular way to use Select Case statements is with a drop down box. You can then test which item a user selected from a list of available items. You're going to write a little programme to do just this. But before you can do so, you'll need to know how to add a Combo Box to a form, and how to get at the values in its list. We'll do that in the next section.

**Add a Combo Box to a VB .NET form**

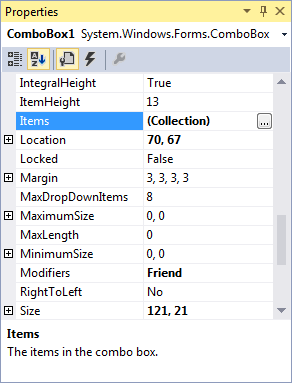
Create a new project for this section. Add a button to your new form. Then, locate the Combo Box on the Visual Basic .NET toolbar. It looks like this:



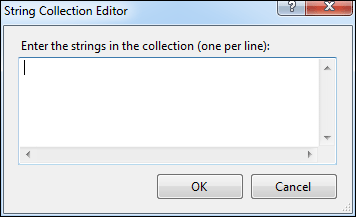
Double click the icon to add a Combo Box to your form. Or click once with the left hand mouse button, and then draw one on the form.

A combo box is a way to limit the choices your user will have. When a black down-pointing arrow is clicked, a drop down list of items appears. The user can then select one of these options. So let's set it up to do that.

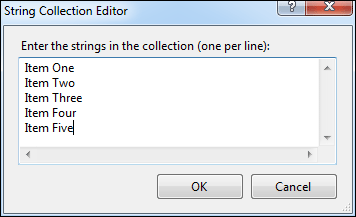
* Click on your Combo Box to select it. Then locate the Item property from the Properties Box:



* Click the grey button, as above. The one with the three dots in it. When you do, you'll get the following box popping up:

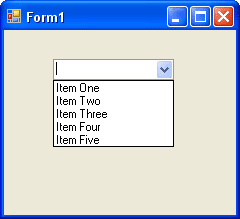


* To use the String Collection Editor, type an item and press Return (it's just like a normal textbox. Each item will be one item in your drop-down box.)
* Enter five items, as in the image below:



* Then click the OK button at the bottom.

The Editor will close, and it will look like nothing has happened. However, run your programme and test out your new Combo Box. You should have something like this:



You now need to know how to get values from the list. Once you know how to get a value from the list, you can put the value into a variable and test it with some Conditional logic.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

Getting a value from a Combo Box is fairly straightforward, because it acts just like a Textbox. A Textbox has a Text property, and so does a Combo Box. To get a value from a Textbox, you would code like this

**MyVariable = Textbox1.Text**

Whatever is in the Textbox will be transferred to the variable called MyVariable. The process is exactly the same for a Combo Box. The code is like this:

**MyVariable = Combobox1.Text**

Now we are transferring whatever is selected from the Combo Box to the variable called MyVariable.

Let's try it. Double click the button you added to your form. This will open the code window. Then enter the following code for the button:

**Dim MyVariable as String**

**MyVariable = Combobox1.Text**

**MsgBox MyVariable**

Run your programme. When the programme is running, select an item from your Combo Box. Then click your button. Whatever was in the Combo Box window should have ended up in the Message Box.

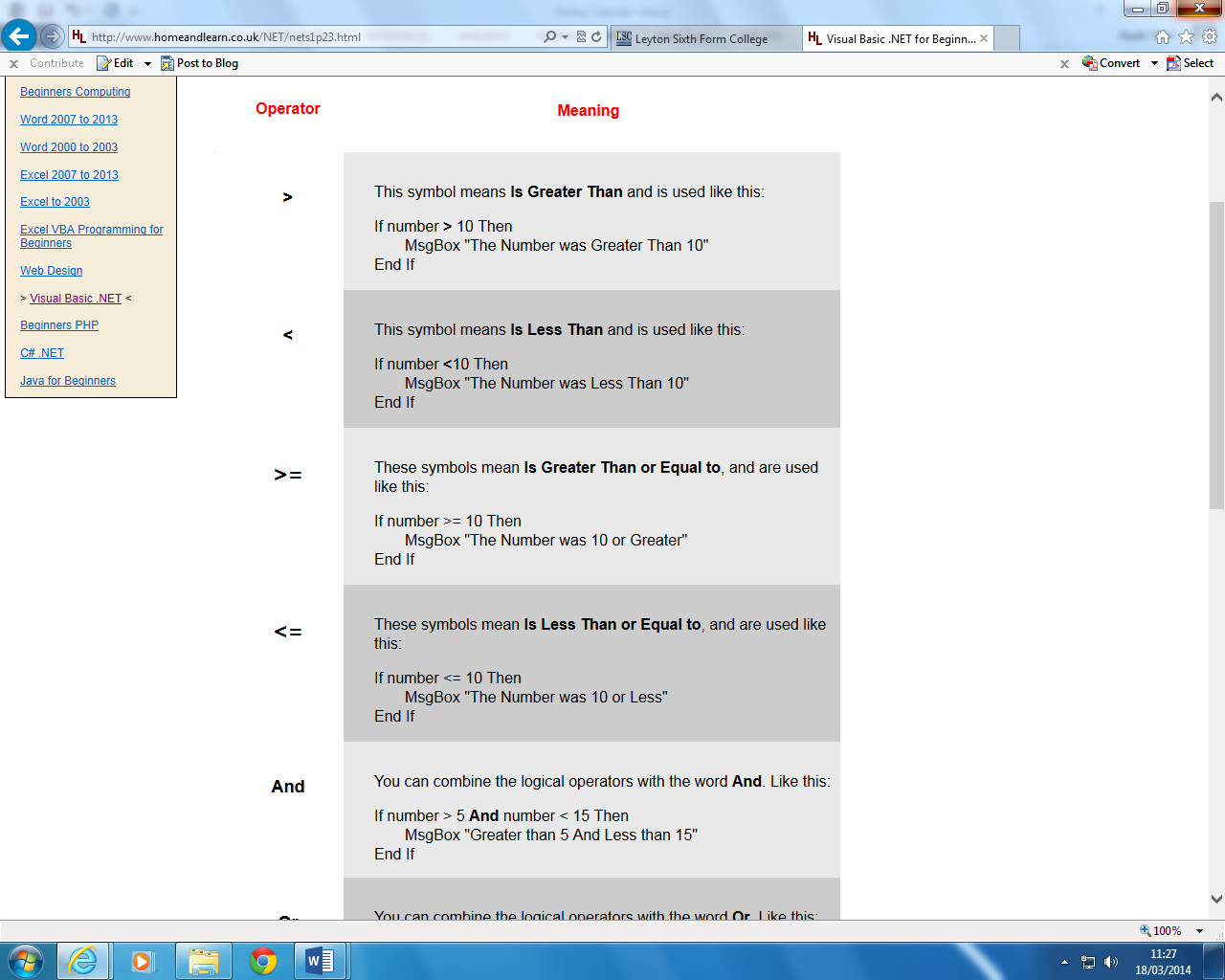
And that's all there is to getting a value from a Combo Box - just access its Text Property and pass it to a variable.

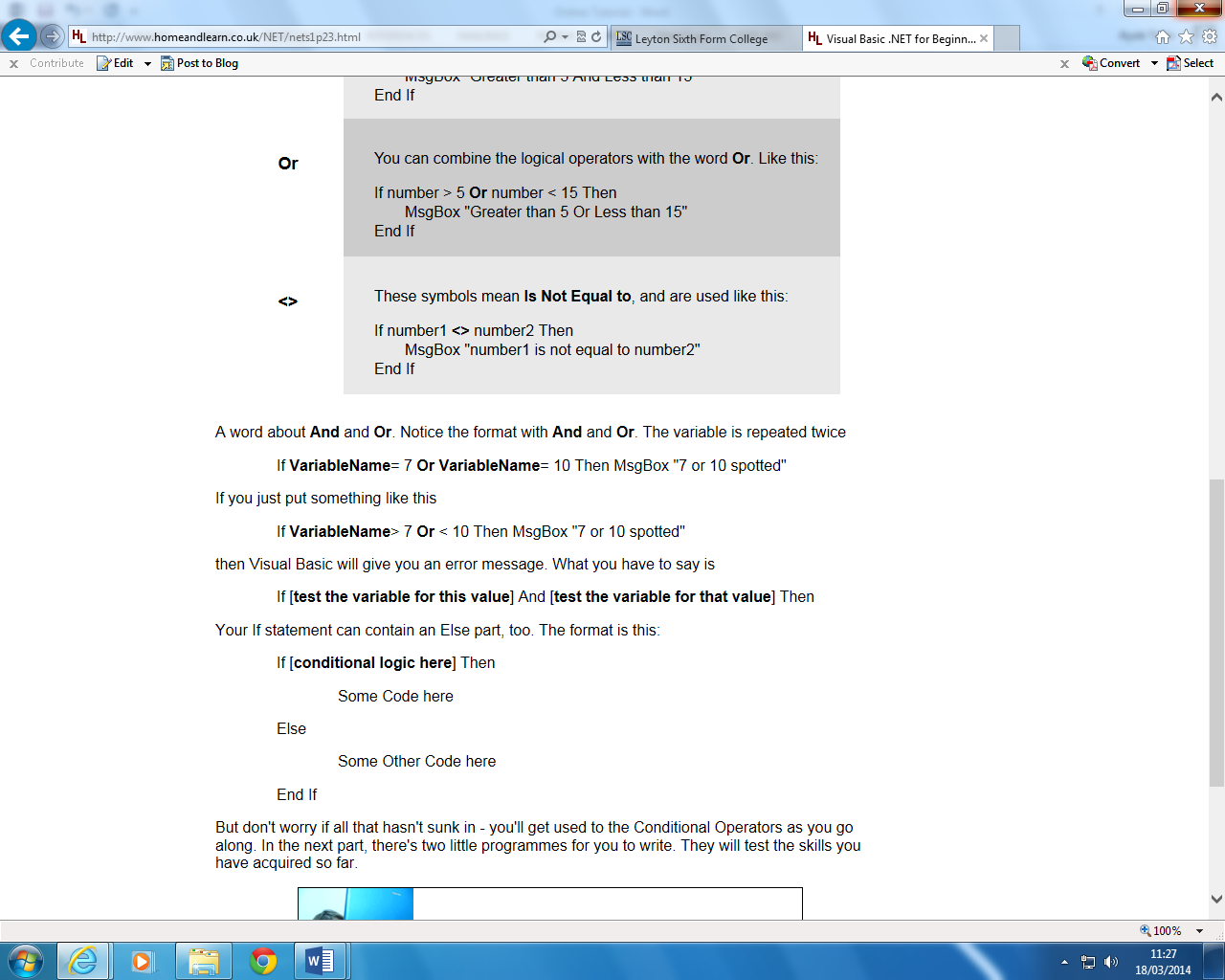
Finally, the Combo Box has a **DropDownStyle** property. Locate this property and you'll notice its value has a drop down box. The box contains three different Combo Box styles to choose from. Experiment with all three and see how they differ.

In the next section, we'll take a look at Conditional Operators, what they are, and how to use them.

# Conditional Operators

The Conditional Operators allow you to refine what you are testing for. Instead of saying "If X is equal to Y", you can specify whether it's greater than, less than, and a whole lot more. Examine the list of Operators:





# Section Three Exercises

### Part 1 - If statements

Start a new project. Add a textbox, a Label and a button to your new Form. Then write a programme that does the following:

1. Asks users to enter a number between 10 and 20.
2. The number will be entered into the Textbox.
3. When the Button is clicked, your Visual Basic code will check the number entered in the Textbox.
4. If it is between 10 and 20, then a message will be displayed.
5. The message box will display the number from the Textbox.
6. If the number entered is not between 10 and 20 then the user will be invited to try again, and whatever was entered in the Textbox will be erased

[Answer to Part One](http://www.homeandlearn.co.uk/NET/vb_answers.html#sec3p1)

### Part 2 - Select Case Statements

Add a Combo box and another button to your form. Create a list of items for your Combo Box. The list of items in your Combo box can be anything you like - pop groups, football teams, favourite foods, anything of your choice. Then try the following:

**Use a select case statement to test what a user has chosen from your drop-down list. Give the user a suitable message when the button was clicked.**

# An Introduction to Loops

There are three types of loop for us to cover with VB.NET: a For loop, a Do loop, and a While … End While loop. This last one is almost the same as a Do loop, and we won't be covering it here. But the other two types of loop come in very handy, and a lot of the time you can't programme effectively without using loops.

### What is a Loop?

A loop is something that goes round and round and round. If someone told you to move your finger around in a loop, you'd know what to do immediately. In programming, loops go round and round and round, too. In fact, they go round and round until you tell them to stop. You can programme without using loops. But it's an awful lot easier with them. Consider this.

You want to add up the numbers 1 to 4: 1 + 2 + 3 + 4. You could do it like this

Dim answer As Integer

answer = 1 + 2 + 3 + 4

MsgBox answer

Fairly simple, you think. And not much code, either. But what if you wanted to add up a thousand numbers? Are you really going to type them all out like that? It's an awful lot of typing. A loop would make life a lot simpler.

But don't get hung up too much on the name of the Loop. Just remember what they do: go round and round until you tell them to stop.

We'll discuss the For Loop first.

# For Loops in VB .NET

We've produced a video to go with this lesson. It's recommended that you read the text below as well, though. The video is here:

[Programming Loops](http://www.homeandlearn.co.uk/NET/video_loops.html)

This first type of loop we'll look at is called a For Loop. It is the most common type of loop you'll use when you programme. We'll use one to add up our 4 numbers, and then discuss the code. Study the following. In fact, create a new Project. Add a button to your new Form. Double click your new button and type the following code for it:

Dim answer As Integer  
Dim startNumber As Integer

answer = 0

For startNumber = 1 To 4

answer = answer + startNumber

Next startNumber

MsgBox answer

Run the programme, and see what happens when you click the button. The number 10 should have been displayed in your message box.

### The For loop code

We start by setting up two integer variables. We set one of these to zero. Then we start our loop code. Let's examine that in more detail.

For startNumber = 1 To 4

answer = answer + startNumber

Next startNumber

We start our loop by telling Visual Basic what type of loop we want to use. In this case it is a **For** loop:

**For** startNumber = 1 To 4

The next thing you have to do is tell Visual Basic what number you want the loop to start at:

For **startNumber = 1** To 4

Here we are saying "Start the loop at the number 1". The variable startNumber can be called anything you like. A popular name to call a start loop variable is the letter i ( i = 1). So what we're doing is setting up a variable - the start of the loop variable - and putting 1 into it;

Next, you have to Tell Visual Basic what number to end the loop on:

For startNumber = 1 **To 4**

The **To** word, followed by a number or variable, tells Visual Basic how many times you want the loop to go round and round. We're telling Visual Basic to loop until the startNumber variable equals 4

The command that tells Visual basic to grab the next number in the sequence is this:

**Next startNumber**

When Visual Basic reaches this line, it checks to see what is in the variable startNumber. It then adds one to it. In other words, "Get me the next number after the one I've just used."

The next thing that happens is that Visual Basic will return to the word **For**. It returns because it's in a loop. It needs to know if it can stop looping. To check to see if it can stop looping, it skips the startNumber = 1 part, and then jumps to your end number. In our case, the end number was 4. Because **Next startNumber** adds one to whatever is in startNumber, then startNumber is now 2 (It was 1 at the start. The next number after one is ... ?).

So if startNumber is now 2, can Visual Basic stop looping? No it can’t. Because we’ve told it to loop until it reaches number 4. It’s only reached number 2, so off it goes on another trip around the loop. When the startNumber is greater than the end number, Visual Basic drops out of the loop and continues on it’s way.

But remember why we're looping: so that we can execute some code over and over again.

To clarify things, change the above code to this:

Dim startNumber As Integer

For startNumber = 1 To 4

MsgBox("Start Number = " & startNumber)

Next startNumber

Run the programme, and click your button. What happens? You should have seen this in the message box, one after the other:

Start Number = 1  
Start Number = 2  
Start Number = 3  
Start Number = 4

Each time round the loop, the code for the message box was executed. You had to click OK four times - startNumber = 1 To 4.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

### Summary

So to sum up:

1. A For loop needs a start position and an end position, and all on the same line
2. A For loop also needs a way to get the next number in the loop
3. A loop without any code to execute looks like this:

For i = startNumber To endNumber

Next i

The above code uses two variables for the start and end numbers. The start number for the loop goes directly into the variable called **i**. When Visual Basic wants the next number, it just add one to whatever is in the variable **i**. You could use it like this:

Dim startNumber As Integer  
Dim endNumber As Integer  
Dim i As Integer

startNumber = 1   
endNumber = 4

For i = startNumber To endNumber

Msgbox i

Next i

Change the code for your button to that new code, and test it out. Study the code so that you understand what is going on.

For Loops might not be easier to understand than just typing **answer = 1 + 2 + 3 + 4**, but they are a lot more powerful if you want to add up a thousand numbers!

### Exercise

Put two textboxes on your form. The first box asks users to enter a start position for a For Loop; the second textbox asks user to enter an end position for the For loop. When a button is clicked, the programme will add up the numbers between the start position and the end position. Display the answer in a message box. You can use this For Loop code

For i = startNumber To endNumber

answer = answer + i

Next i

Get the **startNumber** and **endNumber** from the textboxes.

### Exercise H

Amend your code to check that the user has entered numbers in the textboxes. You will need an If statement to do this. If there's nothing in the textboxes, you can halt the programme with this code

**Exit Sub**

For this exercise, you will be passing whatever is in the textboxes to integer variables. It is these variables you are checking with your If Statement. Because numbers will be entered into the textboxes, remember to convert the text to a value with Val( ).

But the Text property will return a zero if the box is empty. So your If statement will need to check the variables for a value of zero. If it finds a zero, Then you can use the Exit Sub code. The If statement should come first, before the For Loop code.

[Answer to Exercise H (Both exercises above)](http://www.homeandlearn.co.uk/NET/vb_answers.html#exH)

This has been quite a long section, and you may need a breather! Don't worry if you don't understand all that in one sitting. Come back to it, and it will sink in - eventaully! In the next section, we'll take a look at Do Loops.

# Do Loops in VB .NET

We saw with the [For Loop](http://www.homeandlearn.co.uk/NET/nets3p2.html) that a specific number of times to loop was coded into it. We said:

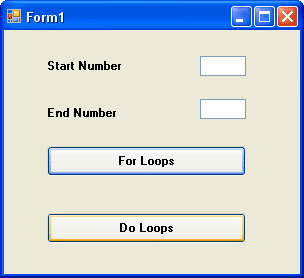
For startNumber = 1 To 4

We knew that we only wanted to loop four times. But what if we don't know how many times around the loop we want to go? Later, we'll be opening text files and reading the data from them. If we used a For loop to get every line of text, we'd have to know before hand how many lines the text file held. A For Loop would not be very efficient in this case.

But a Do Loop would be. With a Do Loop we can use word s like "While" and "Until". And then we can say, "Go round and round the loop While there's still text to be read from the file." An example might make things clearer.

Load the form you created for the last exercise, the one that has two textboxes and a Button and tested your understanding of For loops.

Add another button to the Form. Your form might look something like this:



Double click the new button to open the code window, and then type the following code for the new button:

Dim number as Integer

number = 1

Do While number < 5

MsgBox number

number = number + 1

Loop

When you've finished, run the programme and see what happens. The numbers 1 to 4 should have displayed in your message box.

So the Do loop keeps going round and around. And when it gets to the top, it tests the "While" part - **Do While number is Less Than 5**. It's really answering a Yes or No question: is the number inside the variable called **number** Less Than 5? If it is Less Than 5, go round the loop again. If it's not Less than 5, Visual Basic jumps out of the Loop entirely.

You can add the "While ... " part to the bottom, just after the word "Loop". Like this:

Do

number = number + 1

Loop While number < 5

Try it and see what difference it makes.

None, right? But there is a difference between the two. With the "While ... " part at the bottom, the code above it gets executed at least once. With the code on the first line after the word "Do", the code might not get executed at all. After all, the number inside the variable might already be Greater Than 5. If it is, Visual Basic won't execute the code.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

### Do ... Until

You have another choice for Do Loops - **Do ... Until**.

There's not much difference between the two, but a Do ... Until works like this. Change your Loop code to the following:

Do Until number < 5

MsgBox number

number = number + 1

Loop

Run the code and see what happens.

Nothing happened, right? That's because we "Keep looping UNTIL the number in the variable called number is Less Than 5" The problem is, the number inside the variable is already Less Than 5. And if the number is Less than 5, then the code won't execute - because it has already met the end condition.

Change that Less Than sign to a Greater Than sign, and then test your code again. Now what happens?

The numbers 1 to 5 should have displayed. Again, the loop keeps going round and around testing to see if our end condition is met, in this case Is Greater Than 5. If the condition is met, VB breaks out of the Loop; if not, keep going round.

Change the Greater Than sign to Greater Than or Equal to ( >= ), and test it again. It should now print 1 to 4.

The "Until" part can go at the bottom, just after the word Loop. Like this

Do

MsgBox number

number = number + 1

Loop Until number >= 5

To sum up, use a Do Loop if you don't know what the end number is going to be, otherwise a For Loop might be better.

You're now going to write a programme that uses a For Loop to work out the times table.

# A Times Table Programme

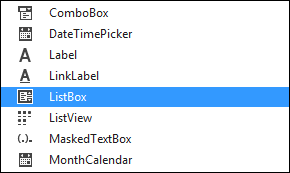
Start a new project for this. Onto your new Form, place two textboxes and a Button. Set the Text property of Textbox1 to 1, and the Text property of Textbox2 to 10. Set the Text property of the Button to "Go".

When the Go button is clicked, the programme will put the numbers from the Textbox into two variables. We'll then put a value into a variable called multiplier. If you're doing the times tables, the format is

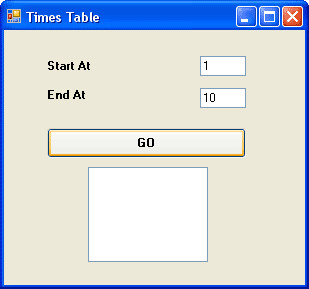
X multiplied by Y = Z  
(2 multiplied by 3 = 6)

We'll use a For Loop to work out the X part; we'll get the Y part from a multiplier variable. We'll then display the results in something called a Listbox.

So add a List Box to your form. It looks like this in the toolbox:



The form you design should look something like this one:



A List box is similar to a Combo Box, in that you have a list of items that the user can select. Here, we're just using it display the results of our programme. We'll add items to the List box with our code, rather than in design time like we did for the Combo box.

So, here's the code for the entire programme. Double click your Go button, and add the following:

Dim number1 As Integer  
Dim number2 As Integer  
Dim multiplier As Integer  
Dim answer As Integer  
Dim i As Integer

number1 = Val(TextBox1.Text)  
number2 = Val(TextBox2.Text)

multiplier = 2

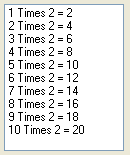
For i = number1 To number2

answer = i \* multiplier

ListBox1.Items.Add(i & " Times " & multiplier & " = " & answer)

Next i

When you've finished, run the programme and see how it works. You should see this appear in your List box:



Let's run through the code to see how it works.We'll do that on the next page. Click the link below to move on.

# The Times Table Code

As you can see from our [last lesson](http://www.homeandlearn.co.uk/NET/nets3p4.html), we've set up five Integer variables - number1, number2, multiplier, answer and i.

The next thing we did was to pass whatever is in our two Textboxes straight into the two variable, number1 and number2. The start number goes into textbox one, and the end number goes into textbox2.

number1 = Val(TextBox1.Text)  
number2 = Val(TextBox2.Text)

In the next line of code, we set a starting value for the multiplier variable:

multiplier = 2

The next thing we had was a For Loop. The For Loop was this:

For i = number1 To number2

answer = i \* multiplier  
ListBox1.Items.Add(i & " Times " & multiplier & " = " & answer)

Next i

Remember: the **number1** and **number2** variables hold our numbers from the Textboxes. We set these to 1 and 10. So our first line of the For Loop is really this:

For i = 1 To 10

We're saying, "Start a For Loop". Whatever is in the variable called number1, make that the starting number for the Loop. Put this value into the variable called i. The end of the Loop will come when the variable called i has the value 10. Stop looping when you reach this value.

The next part of the code reads this:

answer = i \* multiplier

This means, Put into the variable called **answer** the following sum: whatever is in the variable called **i** multiplied by whatever is in the variable called **multiplier**.

[No more reading these lessons online - get the eBook here!](http://www.homeandlearn.co.uk/bookshop/net.html)

Finally, a word about the line that displays your text in the list box. It was this:

**ListBox1.Items.Add(i & " Times " & multiplier & " = " & answer)**

To add items to a list box with code, first you type the name of your list box:

**ListBox1**

Type a full stop and a drop down list will appear. Select Items from the list.

**ListBox1.Items**

Type another full stop and again a drop down list will appear. Select the Add Method

**ListBox1.Items.Add**

This method, not surprisingly, lets you add items to your list box. Whatever you want to add goes between a pair of round brackets:

**ListBox1.Items.Add( )**

In between the round brackets, we have this for our code:

**i & " Times " & multiplier & " = " & answer**

It might be a bit long, but there are 5 parts to it, all joined together by the concatenate symbol (&):

i  
" Times "  
multiplier  
" = "  
answer

The variable **i** holds the current value of the For Loop; " Times " is just direct text; **multiplier** holds the value we're multiplying by (our times table); " = " is again direct text; and **answer** is the answer to our times table sum.

If you want to clear the items from a List box you can do this. At the top of the code, enter this line:

**ListBox1.Items.Clear()**

So instead of selecting Add from the final drop down list, select Clear.

### Exercise

Add another textbox to your form. This will represent the "times table". So far you have been getting this value directly from the code. For this exercise, get the multiplier value directly from the textbox. Add appropriate labels to all your textboxes.

In the next part, we'll take a closer look at the basic Maths symbols you can use in VB .NET.

# The Basic Math Symbols in VB .NET

If you're doing any programming, you need to know how to use the basic Math symbols. The basic Math symbols in Visual Basic .NET are these:

+ The Plus sign adds numbers together

- The minus sign takes one number away from another

\* The asterisk symbol, above the number 8 on your keyboard, is used to multiply

/ The forward slash on your keyboard is the divide by symbol

= The equals sign

A word or two about how to use the mathematical symbols in Visual Basic. You can use the operators by themselves:

answer = 8 + 4  
answer = 8 - 4  
answer = 8 \* 4  
answer = 8 / 4

Or you can combine them by using parentheses.

Here, Visual Basic will work out the sums in parentheses first, and then add the two sums together

answer = (8 - 4) + (4 -2)  
answer = 4 + 2  
answer = 6

But you've got to be careful with parentheses, because there is a strict order that VB uses when it's doing maths. Consider this sum:

answer = 8 - 4 + 4 + 2 \* 2

Try that code behind a new button. Display the result in a MsgBox. What answer did you get? 12! It's wrong! But why?

You would think it would work out the sum like we do - left to right

8 - 4 = 4  
+ 4 = 8  
+ 2 = 10  
\* 2 = 20

But VB doesn't work it out like that. Visual basic will do the multiplying first. So it will calculate like this

2 \* 2 = 4  
8 - 4 + 4 = 8  
8 + 4 = 12

To make sure Visual Basic does your sums correctly you have to be careful of the parentheses. Try changing the code to this:

answer = (8 - 4) + (4 + 2) \* 2

Now what happens. That's right - you get 16! It's still wrong! At least it is if you are working from left to right. But Visual Basic isn't. It will do the (4 + 2) \* 2 part first, and then add that to 8 - 4. Which gives you 16.

In order to force Visual Basic to get the sum right, you need even more parentheses. Try this code and see what happens:

answer= ((8 - 4) + (4 + 2)) \* 2

Finally we get the answer we've been expecting - 20! The parentheses above have grouped our sums into separate sections, thereby forcing VB to do the sums in the right order.

So take care when using parentheses to do your sums: the order that Visual Basic does its sums does matter!

In the next section of the course, we'll have some fun adding menus to a Visual basic .NET form.

…………………………………………………………

# VB NET Exercise Answers

Exercise A (Button code not shown)

Dim number1 As Integer  
Dim number2 As Integer  
Dim number3 As Integer = 10  
Dim answer As Integer

number1 = 3  
number2 = 5

answer = number1 + number2

answer = answer \* number3

Textbox1.Text = answer

Exercise B

Dim FirstName As String  
Dim LastName As String  
Dim FullName As String

FirstName ="Bill"  
LastName = "Gates"

FullName = LastName & " " & FirstName

Textbox1.Text = FullName

Exercise C

Dim FirstName As String  
Dim MiddleName As String  
Dim LastName As String  
Dim FullName As String

FirstName ="Bill"  
MiddleName = "Henry"  
LastName = "Gates"

FullName = FirstName & " " & MiddleName & " " & LastName

Textbox1.Text = FullName

Exercise D

Dim FirstName As String  
Dim LastName As String  
Dim WholeName As String

FirstName = txtFirstName.Text  
LastName = txtLastName.Text

WholeName = FirstName & " " & LastName

txtWholeName.Text = WholeName

Exercise E

Dim ButtonContents As String  
ButtonContents = Button1.Text

txtVariables.Text = ButtonContents

Exercise F

[View the full code for the calculator](http://www.homeandlearn.co.uk/NET/CALCULATOR1.txt)

Exercise G

Dim agerange As Integer

agerange = Val(textbox1.Text)

Select Case agerange

Case 13 To 19

MsgBox “In your Teens”

Case 20 To 29

MsgBox “In your Twenties”

Case 30 To 39

MsgBox “In Thirties ”

Case Else

MsgBox “Not at a teen and over 39”

End Select

Section 3, Part One

Dim number\_in\_textbox As Integer

number\_in\_textbox = Val(TextBox1.Text)

If number\_in\_textbox > 10 And number\_in\_textbox < 20 Then

MessageBox.Show("Number entered was: " & number\_in\_textbox)

Else

MessageBox.Show("Number entered was not between 10 and 20")  
TextBox1.Text = ""

End If

Section 3, Part Two

Item One, Item Two, Item Three below can be replaced by whatever items you have in your Combo Box list.

Dim MyVariable As String

MyVariable = ComboBox1.Text

Select Case MyVariable

Case "Item One"

MsgBox("You like " & MyVariable & ", I see - good choice")

Case "Item Two"

MsgBox("Really? " & MyVariable & "?")

Case "Item Three"

MsgBox(MyVariable & "? Now you're just being silly!")

End Select

Exercise H

Dim startNumber As Integer  
Dim endNumber As Integer  
Dim i As Integer  
Dim answer As Integer

startNumber = Val(TextBox1.Text)  
endNumber = Val(TextBox2.Text)

If startNumber = 0 Or endNumber = 0 Then

MsgBox("Empty textbox")  
Exit Sub

End If

For i = startNumber To endNumber

answer = answer + i

Next i

MsgBox(answer)

Exercise I

Dim message As String = ""  
Dim counter As Integer = 0

If CheckBox1.CheckState = CheckState.Checked Then

message = message & CheckBox1.Text & vbNewLine  
counter = counter + 1

End If

If CheckBox2.CheckState = CheckState.Checked Then

message = message & CheckBox2.Text & vbNewLine  
counter = counter + 1

End If

If CheckBox3.CheckState = CheckState.Checked Then

message = message & CheckBox3.Text & vbNewLine  
counter = counter + 1

End If

If CheckBox4.CheckState = CheckState.Checked Then

message = message & CheckBox4.Text & vbNewLine  
counter = counter + 1

End If

If CheckBox5.CheckState = CheckState.Checked Then

message = message & CheckBox5.Text & vbNewLine  
counter = counter + 1

End If

MessageBox.Show("You have chosen " & vbNewLine & message)

Select Case counter

Case 0

MessageBox.Show("You obviously don’t watch soaps!")

Case 1

MessageBox.Show("only 1 show watched")

Case 2

MessageBox.Show("2 shows watched")

Case 3

MessageBox.Show("3 shows watched")

Case 4

MessageBox.Show("4 shows watched")

Case 5

MessageBox.Show("a soap addict")

End Select

Exercise J

Tricky one, this. The solution is to change this line:

letter = strText.Substring(1)

to this:

letter = strText.Substring(i)

By using the loop variable i you advanced the counter for Substring.

Exercise K

Dim OneCharacter As Char  
Dim FirstName As String  
Dim i As Integer  
Dim TextLength As Integer

FirstName = txtChars.Text  
TextLength = FirstName.Length

For i = 0 To TextLength - 1

OneCharacter = FirstName.Chars(i)

If IsNumeric(OneCharacter) Then

MessageBox.Show("Number Found - Exiting")  
Exit Sub

End If

Next

Exercise L

Dim OneCharacter As Char  
Dim FirstName As String  
Dim i As Integer  
Dim TextLength As Integer  
Dim numbercount As Integer = 0

FirstName = txtChars.Text  
TextLength = FirstName.Length

For i = 0 To TextLength - 1

OneCharacter = FirstName.Chars(i)

If IsNumeric(OneCharacter) Then

numbercount = numbercount + 1

End If

Next

MsgBox("found " & numbercount & " numbers")

Exercise M

Checking for a valid email address is quite tricky. In the code below, the Sub is just checking if the email address has an @ sign. Notice that the Sub uses two paramaters, one to pass in the email address and one to pass in the @ sign. You can then reuse this code to pass in, say, an email address and the characters ".com"

'===================================  
' BUTTON CODE  
'===================================  
Dim email As String  
Dim chars\_to\_check As Char

email = txtEmail.Text  
chars\_to\_check = "@"

TestEmail1(email, chars\_to\_check)

'===================================  
' SUBROUTINE  
'===================================  
Private Sub TestEmail1(ByVal eMail As String, ByVal CheckChars As String)

Dim position As Integer

position = InStr(eMail, CheckChars)

If position = 0 Then

MsgBox("Not a Valid email address")

End If

End Sub

Exercise N

Private Sub TextBox1\_Leave(ByVal sender As Object, ByVal e As System.EventArgs) Handles TextBox1.Leave

Dim CheckTextBox As String

CheckTextBox = Trim(TextBox1.Text)  
CheckTextBox = StrConv(CheckTextBox, VbStrConv.ProperCase)  
TextBox1.Text = CheckTextBox

End Sub

Exercise O

Here are the Set and Get properties that change the height and width. (As was mentioned in the text, however, the name of the Class (ConvertPostcode) is not a very good name for what this does - changing the height and width of an image! Plus, picture boxes already have a height and width property)

Private intHeight As Integer  
Private intWidth As Integer

Public Property ChangeHeight() As Integer

Get

Return intHeight

End Get

Set(ByVal HeightValue As Integer)

intHeight = HeightValue

End Set

End Property

Public Property ChangeWidth() As Integer

Get

Return intWidth

End Get

Set(ByVal WidthValue As Integer)

intWidth = WidthValue

End Set

End Property

And here is the code for the button that uses the two properties:

Dim objAlterPicPox As ConvertPostcode  
Dim NewHeight As Integer  
Dim NewWidth As Integer

objAlterPicPox = New ConvertPostcode

objAlterPicPox.ChangeHeight = Val(txtHeight.Text)  
NewHeight = objAlterPicPox.ChangeHeight  
PictureBox1.Height = NewHeight

objAlterPicPox.ChangeWidth = Val(txtWidth.Text)  
NewWidth = objAlterPicPox.ChangeWidth  
PictureBox1.Width = NewWidth

objAlterPicPox = Nothing