

IENG314 / MANE314 Operations Research – II

VBA – Session 3

Review Example

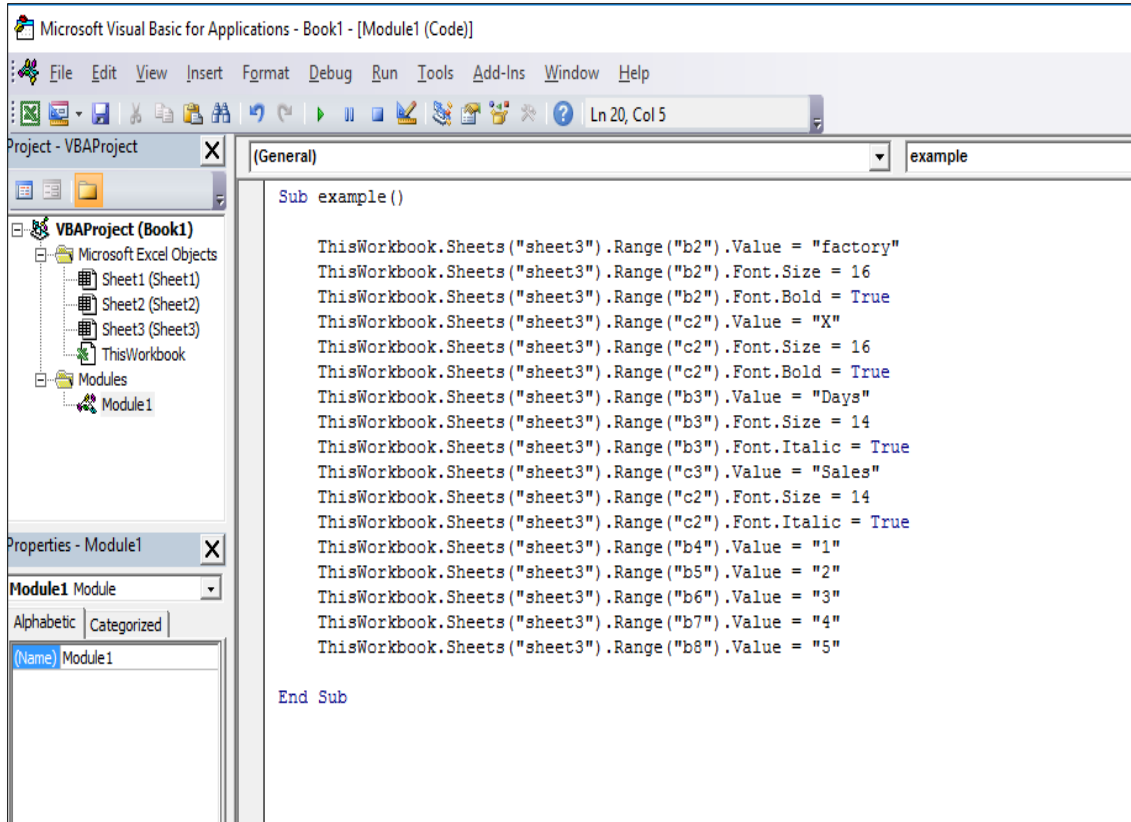
Do the following table in **sheet 3** from cells **B2** to **C8** by using coding in VBA module.

Factory	X
<i>Days</i>	<i>Sales</i>
1	
2	
3	
4	
5	

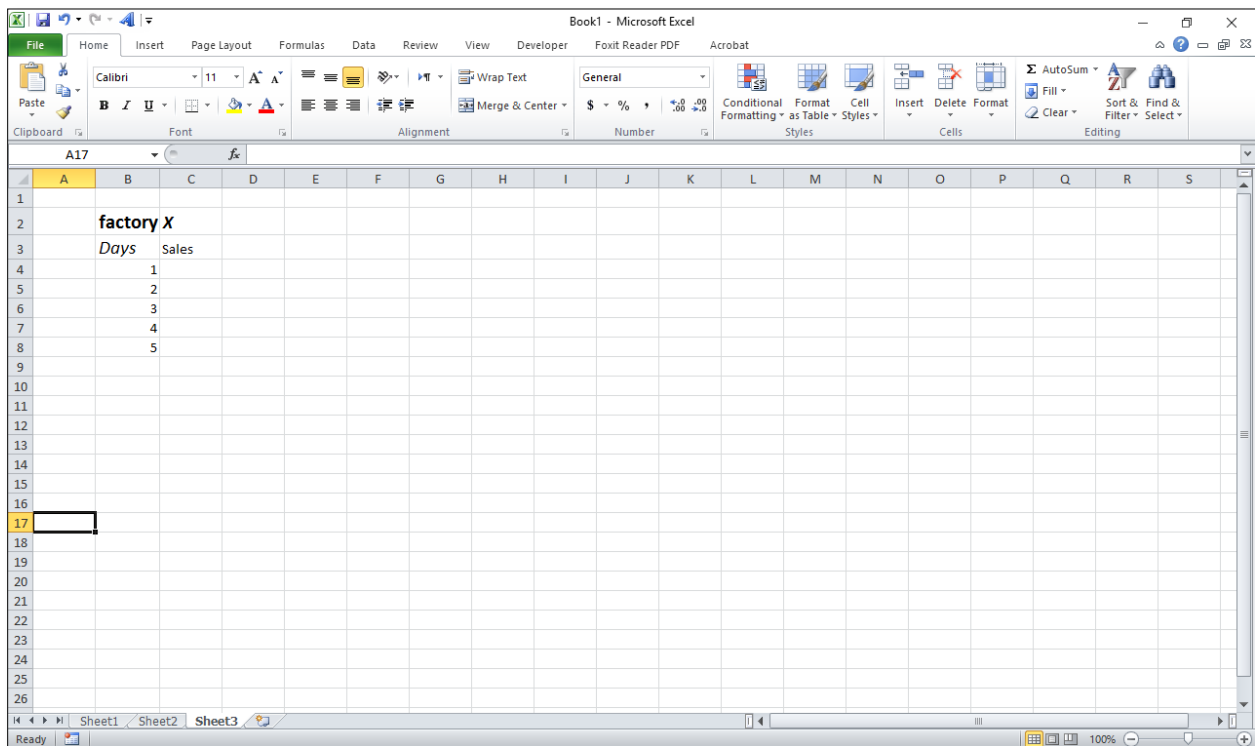
Factory , X : font size=16, font type is bold

Days, Sales : font size=14, font type is italic

For having this table in **sheet 3**, your codes should be like this:



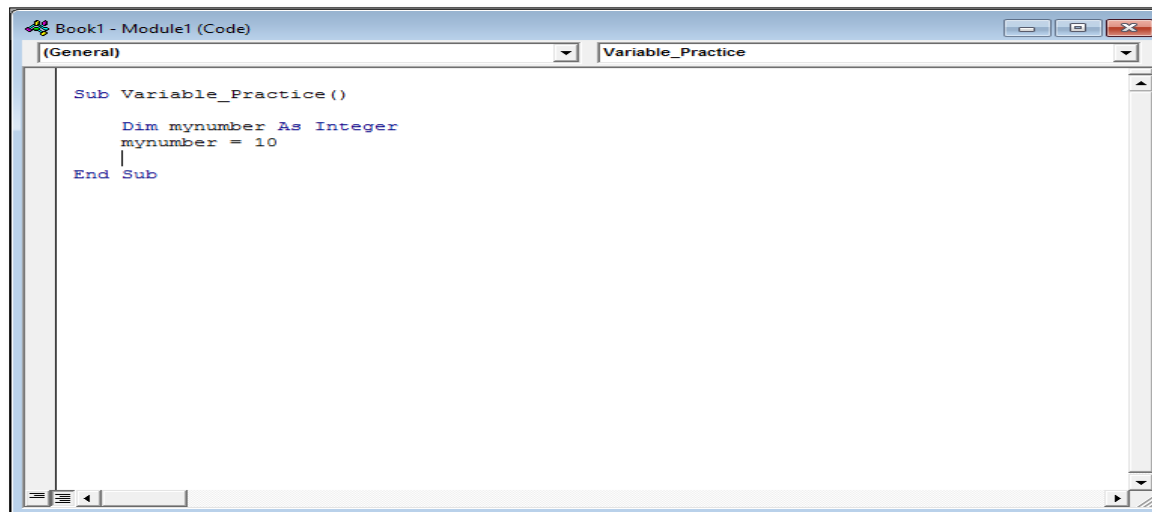
Your result will appear in **sheet 3** in cells from **B2** to **C8**.



VBA Programming Variables

In computer programming, you need to store things in memory, things like numbers and strings of text. You store them so that you can retrieve them later and manipulate the numbers or strings of text in some way. To store things in memory you use a **variable**. Variables have a name and a type.

This is done mostly with the word **Dim**. For example:



```
Book1 - Module1 (Code)
(General) Variable_Practice

Sub Variable_Practice()
    Dim mynumber As Integer
    mynumber = 10
End Sub
```

The above code sets up a variable with the name **MyNumber**. The type of variable is defined by **As Integer**. The above line means: Assign a value of 10 to the variable called **MyNumber**.

Variable Names Rules

You can call your variable just about anything you like. But there are a few things you're not allowed to do. They are:

- You can't start a variable name with a number
- You can't have spaces in your variable names, or full stops (periods)
- You can't use any of the following characters: !, %, ?, #, \$

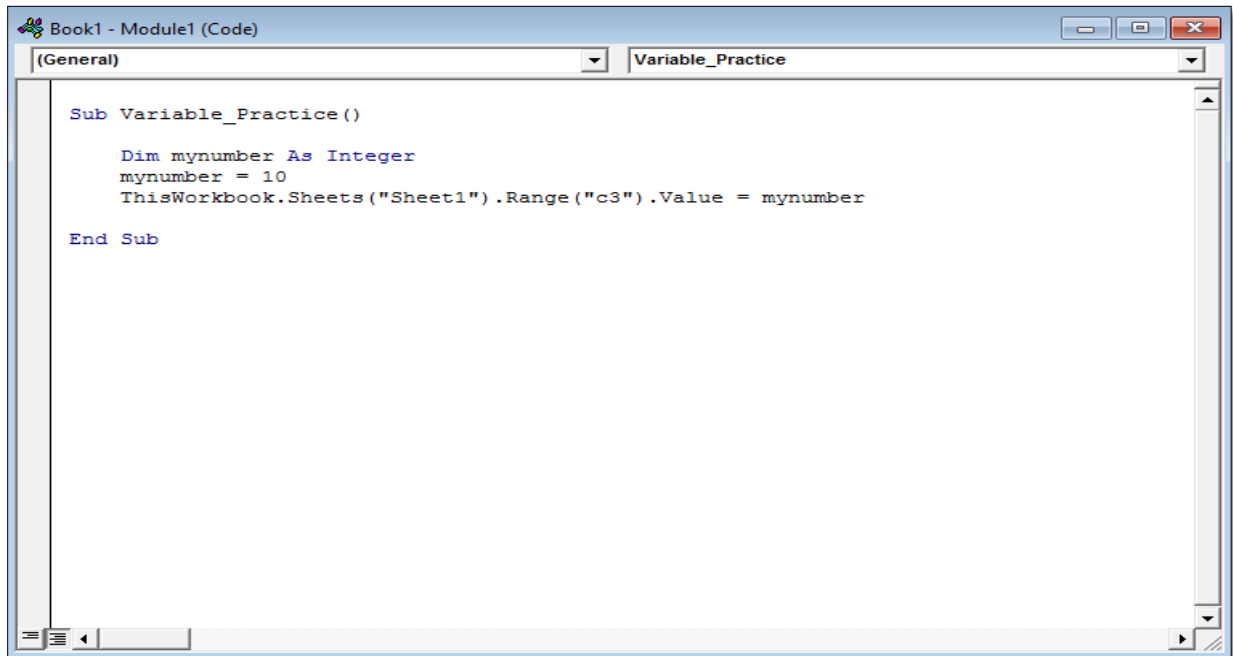
So these variable names are OK

MyVariable
My_Variable
myvariable2

But these variable names will get you an error:

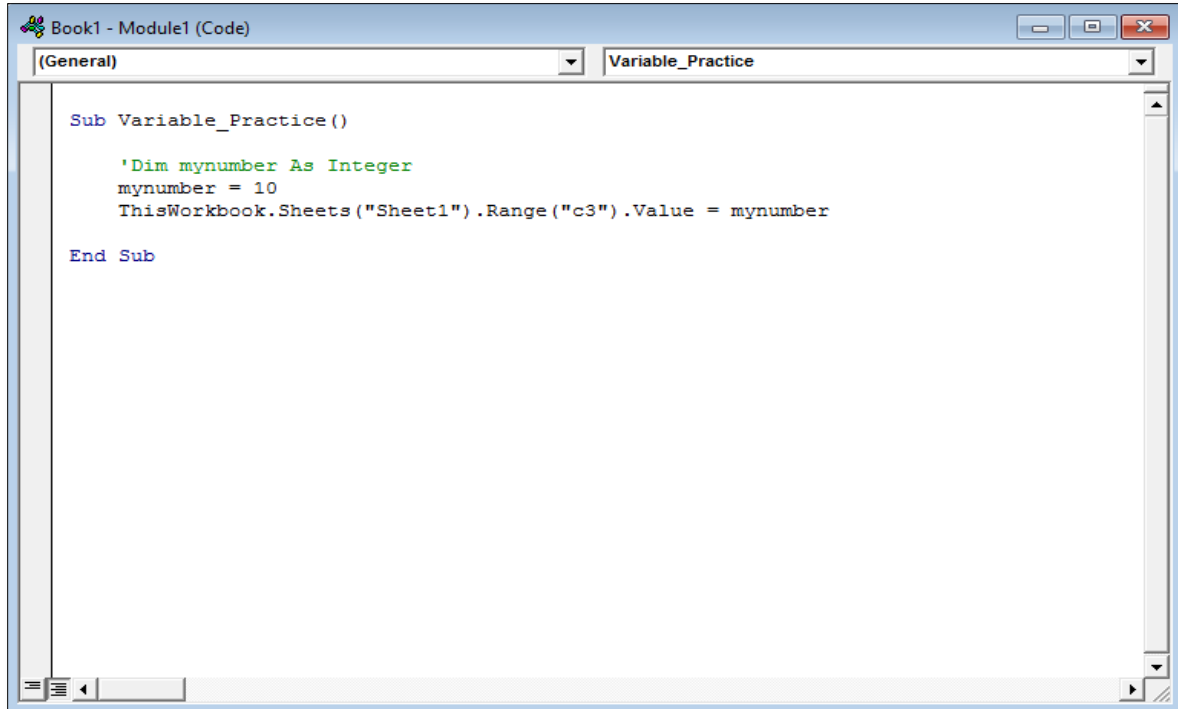
2MyVariable
My Variable
\$myvariable

Transfer our stored value to a cell on a spreadsheet.



```
Sub Variable_Practice()  
    Dim mynumber As Integer  
    mynumber = 10  
    ThisWorkbook.Sheets("Sheet1").Range("c3").Value = mynumber  
End Sub
```

Comment the line out by typing a single quote mark (') before it. Your code will then look like this:



```
Book1 - Module1 (Code)
(General) Variable_Practice

Sub Variable_Practice()
    'Dim mynumber As Integer
    mynumber = 10
    ThisWorkbook.Sheets("Sheet1").Range("c3").Value = mynumber
End Sub
```

A comment means that the line will be ignored. So if you don't need to set up variables with the **Dim** keyword, why bother using them at all? It will cause your code to run really slowly compared to setting up variables with the **Dim** keyword and using a named type like **As Integer**.

Option Explicit

In order to ensure that you don't have any variables that start with the Dim keyword, you can type **Option Explicit** at the very top of your coding window. For example:

The screenshot shows the VBA Code Editor window titled "Book1 - Module1 (Code)". The "General" tab is selected, and the code is in a module named "test". The code is as follows:

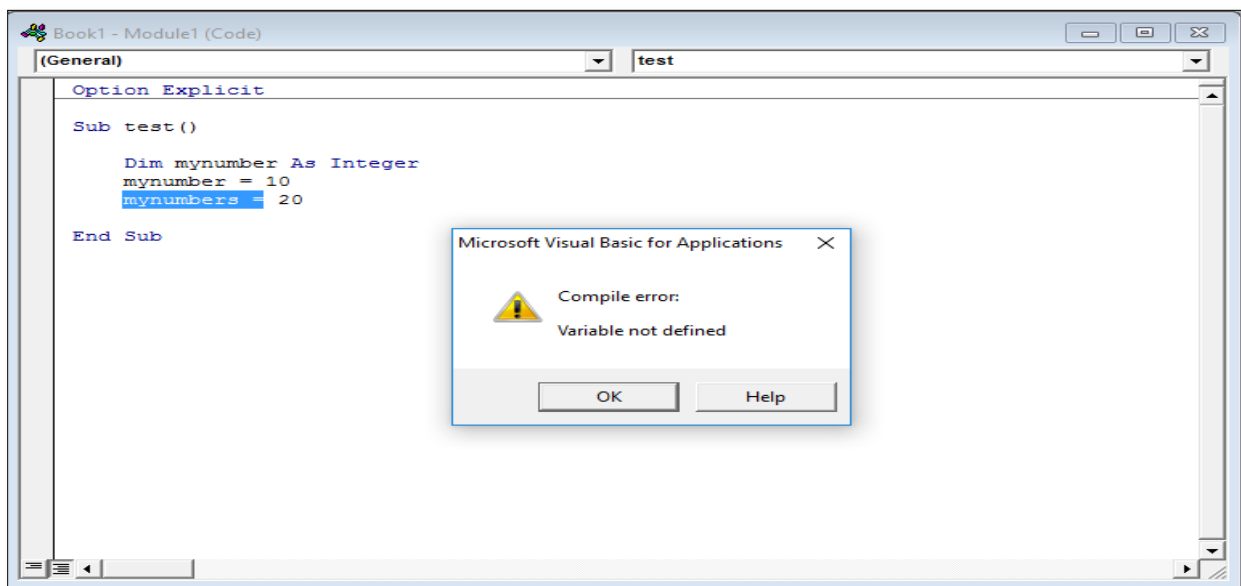
```
Option Explicit

Sub test()

    Dim mynumber As Integer
    mynumber = 10
    mynumbers = 20

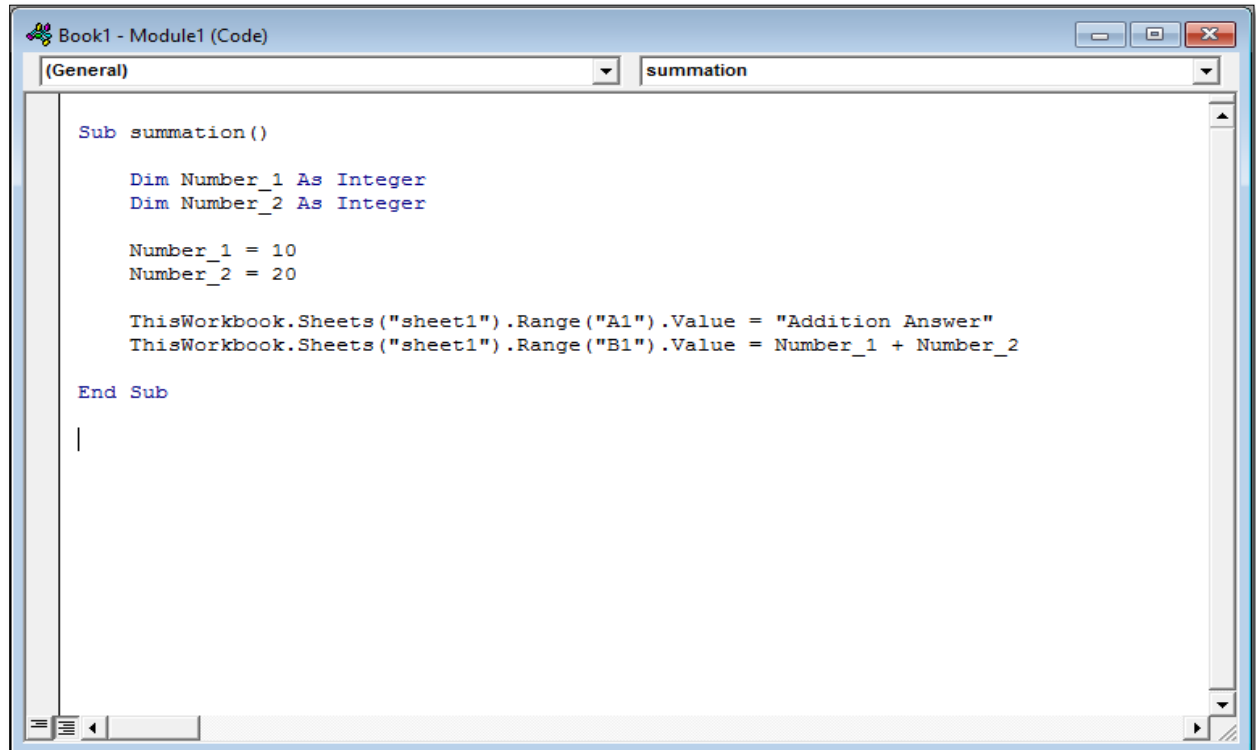
End Sub
```

We have accidentally typed **MyNumbers** instead of **MyNumber**. If we didn't have **Option Explicit** at the top then VBA would have run this code and set up **MyNumbers** as a new variable. This could have led to errors further down the code, if we didn't realize that **MyNumber** and **MyNumbers** were two different variables. Instead, VBA will now not run the code at all. We'll get this error message:



Basic math operations

When you store numbers inside of variables, one of the things you can do with them is mathematical calculations. We'll start with simple addition.



```
Book1 - Module1 (Code)
(General) summation

Sub summation()

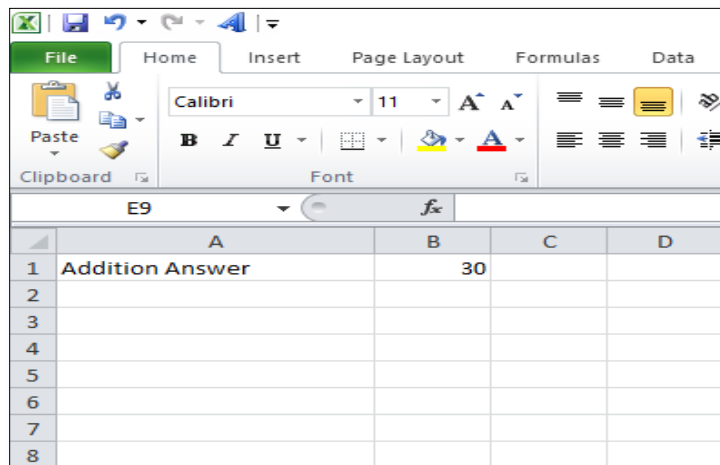
    Dim Number_1 As Integer
    Dim Number_2 As Integer

    Number_1 = 10
    Number_2 = 20

    ThisWorkbook.Sheets("sheet1").Range("A1").Value = "Addition Answer"
    ThisWorkbook.Sheets("sheet1").Range("B1").Value = Number_1 + Number_2

End Sub
```

The result should be this.



	A	B	C	D
1	Addition Answer	30		
2				
3				
4				
5				
6				
7				
8				

In the VBA programming language, the minus sign (-) is used to subtract one value from another.

```
Book1 - Module1 (Code)
(General) subtraction

Sub summation()

    Dim Number_1 As Integer
    Dim Number_2 As Integer

    Number_1 = 10
    Number_2 = 20

    ThisWorkbook.Sheets("sheet1").Range("A1").Value = "Addition Answer"
    ThisWorkbook.Sheets("sheet1").Range("B1").Value = Number_1 + Number_2

End Sub

Sub subtraction()
    Dim Number_1 As Integer
    Dim Number_2 As Integer

    Number_1 = 10
    Number_2 = 7

    ThisWorkbook.Sheets("sheet1").Range("A2").Value = "Subtraction Answer"
    ThisWorkbook.Sheets("sheet1").Range("B2").Value = Number_1 - Number_2

End Sub
```

You should see a new line appear on your spreadsheet:

	A	B	C	D
1	Addition Answer	30		
2	Subtraction Answer	3		
3				
4				
5				
6				
7				
8				

In VBA, the multiplication sign is the asterisk (*).


```
Book1 - Module1 (Code)
multiplication

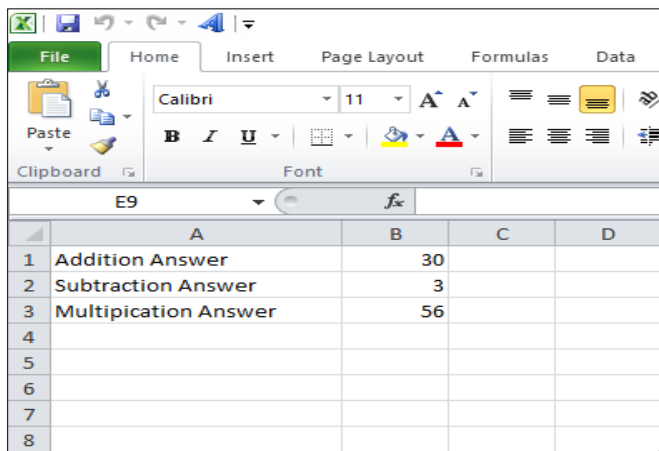
ThisWorkbook.Sheets("sheet1").Range("B2").Value = Number_1 - Number_2
End Sub

Sub multiplication()
    Dim Number_1 As Integer
    Dim Number_2 As Integer

    Number_1 = 8
    Number_2 = 7

    ThisWorkbook.Sheets("sheet1").Range("A3").Value = "Multiplication Answer"
    ThisWorkbook.Sheets("sheet1").Range("B3").Value = Number_1 * Number_2
End Sub
```

You should see a new line added



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D
1	Addition Answer	30		
2	Subtraction Answer	3		
3	Multiplication Answer	56		
4				
5				
6				
7				
8				

The symbol to use when you want to divide numbers is the forward slash (/).

```
Book1 - Module1 (Code)
(General) | division

Dim Number_2 As Integer

Number_1 = 8
Number_2 = 7

ThisWorkbook.Sheets("sheet1").Range("A3").Value = "Multiplication Answer"
ThisWorkbook.Sheets("sheet1").Range("B3").Value = Number_1 * Number_2

End Sub

Sub division()

Dim Number_1 As Integer
Dim Number_2 As Integer

Number_1 = 45
Number_2 = 9

ThisWorkbook.Sheets("sheet1").Range("A4").Value = "Division Answer"
ThisWorkbook.Sheets("sheet1").Range("B4").Value = Number_1 / Number_2

End Sub
```

You should see a new line appear:

The screenshot shows the Microsoft Excel interface with the following data in the worksheet:

	A	B	C	D
1	Addition Answer	30		
2	Subtraction Answer	3		
3	Multiplication Answer	56		
4	Division Answer	5		
5				
6				
7				
8				

With **operator precedence** you have to take into account the following:

- Division and Multiplication are done before addition and subtraction
- Division and Multiplication have equal priority and so are calculated from left to right, as long as there's no addition and subtraction to do
- Addition and subtraction have equal priority and so are calculated from left to right, as long as there's no division and multiplication to do