Conditional Logic

Select...Case Statement (Visual Basic)

One of the best methods for multiple conditions is the Select Case Statement. It does away with the need for multiple If Statements which can make Excel VBA code very hard to read and decipher. The Select Case Statement has **the structure**:

```
Select Case < Expression to test>
   Case (condition)
     Do something
   Case Else
     Do something else
End Select
```

As you can see the **Select Case Statement** is very similar to the **If Statement** in that it will only perform some action if/when a condition is met. However, as you will learn, the Select Case is far more flexible. Let's look at the "Select Case" Statement in it's simplest form.

```
Sub TheSelectCase1()
   Select Case Range("A1").Value
        Case 100
           Range("B1") = 50
   End Select
End Sub
```

Now let us say you need to perform any one of 5 actions depending on the Value of Range A1. If so we could use.

```
Sub TheSelectCase2()
   Select Case Range("A1").Value
      Case 100
         Range("B1").Value = 50
      Case 150
         Range("B1").Value = 40
      Case 200
         Range("B1").Value = 30
      Case 350
         Range("B1").Value = 20
      Case 400
         Range("B1").Value = 10
   End Select
End Sub
```

This, in my opinion, is a far better structure and easier to read than an **If Statement** with multiple **ElseIf** Statements. If none of the above Conditions were met, nothing would occur, unless we use the **optional Case Else Statement**, like:

```
Sub TheSelectCase3()
   Select Case Range("A1").Value
      Case 100
         Range("B1").Value = 50
      Case 150
         Range("B1").Value = 40
      Case 200
         Range("B1").Value = 30
      Case 350
         Range("B1").Value = 20
      Case 400
         Range("B1").Value = 10
      Case Else
         Range("B1").Value = 0
   End Select
End Sub
```

So if the value of range A1 is **NOT** 100,150,200,350 or 400 then place a Value of 0 (zero) in Range B1.

Now while this demonstrates how we can check multiple conditions with the **Select Case Statement**, what if we want to perform some action if the Range A1 is equal to any one of the Values 100,150,200,350 or 400. If this is the case (no pun intended) we could use:

```
Sub TheSelectCase4()
   Select Case Range("A1").Value
      Case 100, 150, 200, 350, 400
         Range("B1").Value = Range("A1").Value
      Case Else
         Range("B1").Value = 0
   End Select
End Sub
```

Let's assume that you only want to perform some action if the range(s) you are checking are between 2 numbers. If this is the case (excuse the pun) then you could use:

```
Sub TheSelectCase5()
   Select Case Range("A1").Value
      Case 100 To 500
         Range("B1").Value = Range("A1").Value
      Case Else
         Range("B1").Value = 0
    End Select
End Sub
```

In this instance, if the range A1 contains a number => 100 and =< 500 then cell A1 value will be placed into cell B1.

Now, what about if we needed to check if cell A1 was not only between 100 and 500 but also between 700 and 1000, 1500 and 2000? No problem, with the Select Case you would simply use:

```
Sub TheSelectCase6()
   Select Case Range("A1").Value
      Case 100 To 500, 700 To 1000, 1500 To 2000
         Range("B1").Value = Range("A1").Value
      Case Else
         Range("B1").Value = 0
   End Select
End Sub
```

In other words if cell A1 contains a number (eg 600) that does not meet the Case criteria, B1 will equal 0. By-the-way, the limit using a Select Case like this is far from 3 criteria!

Example:

Instead of multiple If Then statements in Excel VBA, you can use the Select Case structure.

Situation:

	A	B	C	D	E
1	70				
2					
3					
4					
5					

Add the following code lines:

1. First, declare two variables. One variable of type Integer named score and one variable of type String named result.

```
Dim score As Integer, result As String
```

2. We initialize the variable score with the value of cell A1.

```
score = Range("A1").Value
```

3. Add the Select Case structure.

```
Select Case score
    Case Is >= 80
        result = "very good"
    Case Is >= 70
        result = "good"
    Case Is >= 60
        result = "sufficient"
    Case Else
        result = "insufficient"
End Select
```

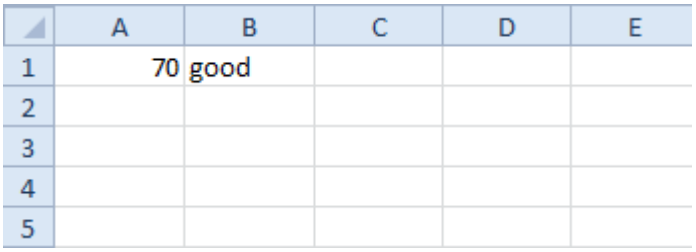
Explanation: Excel VBA uses the value of the variable score to test each subsequent Case statement to see if the code under the Case statement should be executed.

4. Write the value of the variable result to cell B1.

```
Range("B1").Value = result
```

5. Test the program.

Result when you click the command button on the sheet:



	A	B	C	D	E
1	70	good			
2					
3					
4					
5					

Note: Excel VBA executes the code under the second Case statement for all values greater than or equal to 70 and less than 80.

Example -Tax Rates

Below we will look at a program in Excel VBA that calculates the tax on an income. The following tax rates apply to individuals who are residents of Australia.

Taxable income	**Tax on this income**
0 - $6,000	Nil
$6,001 - $35,000	15c for each $1 over $6,000
$35,001 - $80,000	$4,350 plus 30c for each $1 over $35,000
$80,001 - $180,000	$17,850 plus 38c for each $1 over $80,000
$180,001 and over	$55,850 plus 45c for each $1 over $180,000

1. First, we declare two double variables. One double variable we call income, and one double variable we call tax.

```
Dim income As Double
Dim tax As Double
```

2. We initialize the variable income with the value of cell A2 and round it.

```
income = Round(Range("A2").Value)
```

3. We place the rounded value into cell A2 again.

```
Range("A2").Value = income
```

4. We use the Select Case statement to calculate the tax on an income. Excel VBA uses income to test each subsequent Case statement to see if the code under the Case statement should be executed.

```
Select Case income
   Case Is >= 180001
      tax = 55850 + 0.45 * (income - 180000)
   Case Is >= 80001
      tax = 17850 + 0.38 * (income - 80000)
   Case Is >= 35001
      tax = 4350 + 0.3 * (income - 35000)
   Case Is >= 6001
      tax = 0.15 * (income - 6000)
   Case Else
      tax = 0
End Select
```

Example: if income is 37000, tax equals 4350 + 0.3 * (37000-35000) = 4350 + 600 = $4950

5. We write the value of the variable tax to cell B2.

```
Range("B2").Value = tax
```

6. Place this code in a command button and test it.

Result:

	A	B	C
1	Taxable income	Tax on this income	
2	37000	4950	
3			
4			
5			

Insert/ Delete Rows in Excel Worksheet using VBA

VBA insert rows excel macro helps while automating and dealing with the records. For example, we may automate certain task based on the number of items in certain category. And the number of items may not be equal in all the situations it may vary time to time. We will see a practical example in this topic.

Example:

The following example will show you how to insert a row in Excel Worksheet. You can insert multiple rows at a time. The first code will Insert a Row at at Row 2 and the second will Insert 3 Rows from 3 to 5.

```
Sub InsertingRows()
Range("A2").EntireRow.Insert


Rows("3:5").EntireRow.Insert
End Sub
```

Excel VBA code to Delete Entire Row example will help us to delete rows in excel worksheet. We can use Delete method of Rows to delete the Entire Row.

The following Excel VBA delete entire row macro code is to delete entire row from the worksheet. This VBA delete entire row macro will delete row which we have mentioned in the code.

```
Sub Delete_EntireRow ()
Rows(1).EntireRow.Delete
End Sub
```

Insert/Delete Columns in Excel Worksheet using VBA

When we are automating any task we may required inserting columns between other columns or left/right side of a column using Excel VBA.

Example

The following example will show how to insert columns in excel worksheets. In this example I am inserting a column at B and inserting multiple columns at C and D.

```
Sub InsertingColumns()
Range("B1").EntireColumn.Insert

Range("C:D").EntireColumn.Insert
End Sub
```

Excel VBA code to Delete Entire Column example will help us to delete Columns in excel worksheet. We can use Delete method of Columns to delete the Entire Column. The following Excel VBA macro code is to delete entire Column from the worksheet. This VBA macro will delete the Column which we have mentioned in the code.

```
Sub Delete_EntireColumn ()
Columns(1).EntireColumn.Delete
End Sub
```

Hide UnHide Rows in Excel Worksheet using VBA

We are required to Hide-UnHide the rows in some types of requirements. For examples we may have data for 3 different categories of items and we may want to show the respective items based on selection. In this case we can achieve by Hiding or Un-hiding the Rows.

You can use EntireRow.Hidden property of Row. If you set hidden property TRUE, it will hide the rows. Or if you set it to FALSE then it will make rows to visible.

Example

The following example will show you how to Hide and Unhide the rows in excel worksheet using VBA. We can Hide or Unhide the multiple rows at a time. In this example I am hiding and Unhiding Rows 5 to 8.

```
Sub  HidingUnHideRows()
Rows("5:8").EntireRow.Hidden = True


Rows("5:8").EntireRow.Hidden = False
End Sub
```

Hide UnHide Columns in Excel Worksheet using VBA

We may need to Hide UnHide the Columns in Excel for some types of requirements. For examples we may have data for 3 different categories and want to show it based on the user selection. We can achieve this by Hiding or Un-hiding the Columns in Excel.

We can use EntireColumn.Hidden property of a Column. You can set the property value to TRUE if you want to hide, Set to FALSE if you want to un-hide the Columns.

The following example will show you how to hide and unhide the Columns in Excel using VBA. In this example I am hiding the Columns B,C and D by setting the Hidden property as TRUE. And then un hiding by setting the Hidden=FALSE.

```
Sub  HidingUnHideColumns()
Columns("B:D").EntireColumn.Hidden = True
Columns("B:D").EntireColumn.Hidden = False
End Sub
```

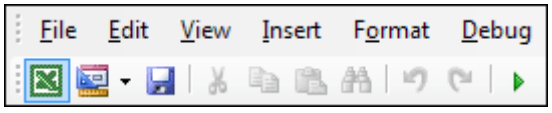
Adding a Button to an Excel Spreadsheet

If you created a simple Sub in the code window, you can activate that Sub from a button on a spreadsheet.

Assume you want to select cells from A1 To D6 by using VBA.

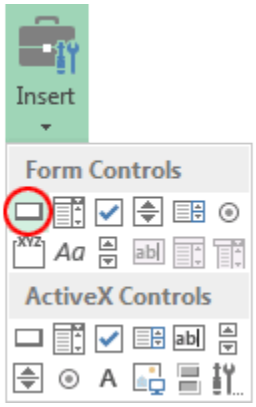
```
Sub Range_A1_D6()
    'Range("A1:D6").Select
End Sub
```

At the top of the VBA Editor, locate the Excel icon, just under the File menu:



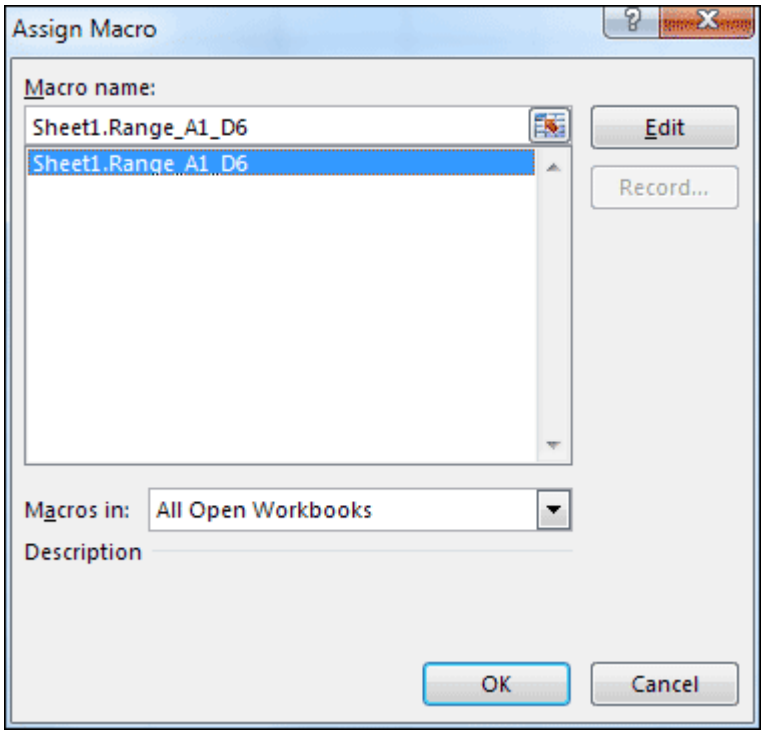
Click this icon to return to your spreadsheet. We'll now place a button control on the spreadsheet.

Locate the **Controls** panel on the **Developer** toolbar, and then click the **Insert** item. From the Insert menu, click the first item, which is a button:

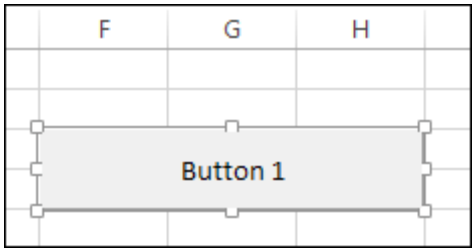


Now move your mouse to your spreadsheet. Hold down your left mouse button somewhere on the F column (F3 will do). Keep it held down and draw out a rectangular button. Let go of the left mouse button when your cursor is on H4

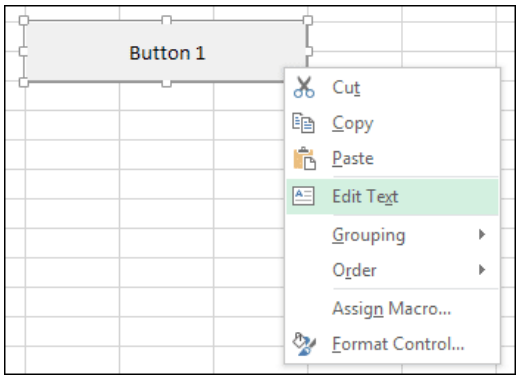
As soon as you let go of the left mouse button you'll see the **Assign Macro** dialogue box appear:



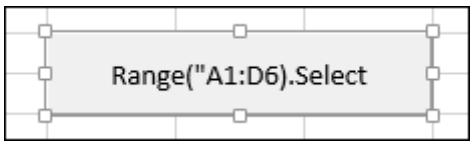
Select your Macro from the list and click OK. The button on your spreadsheet should now look like this:



You can edit the text on a button quite easily. Right click the button to see a menu appear. From the menu, select **Edit Text**:

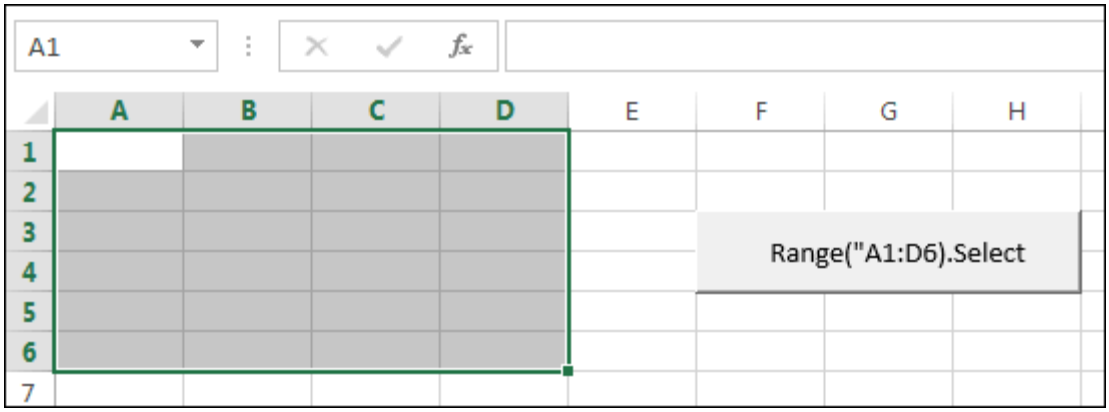


When you select **Edit Text**, a cursor will appear at the start of the text. Use the arrow keys on your keyboard to move the cursor to the end of the line. Delete the text **Button 1** and type **Range("A1:D6").Select** instead (If you accidentally click away from the button, click on it again with right mouse button and not the left button. This will select your button again.):



Click away from the button to exit edit mode and you'll see the sizing handles disappear.

You can now test your button out. Give it a click and you'll see the cells A1 to D6 highlighted:


A1
A
B
C
D
E
F
G
H
1
2
3
4
5
6
7
Range("A1:D6).Select