Excel VBA - Cells Property

As well as referring to cells on a spreadsheet with **Range** you can use **Cells**. The Cells property has an **Item** property that you use to reference the cells on your spreadsheet:

```
Cells.Item(Row, Column)
```

The Row is always a number. But the column can be a number or letter:

```
Cells.Item(1, 1)

Cells.Item(1, "A")
```

Both the lines above refer to the cell A1.

You can shorten this even further and get rid of the **Item** property altogether:

```
Cells(1, 1)

Cells(1, "A")
```

The reason why we're discussing the Cells property is because it's very useful in programming loops. That's because you can replace the numbers between round brackets with a value from your loop. Let's clear that up.

Create another Sub in your coding window from the previous section. Call it **CellsExample**. Add the following code:

```
Dim StartNumber As Integer
Dim EndNumber As Integer

EndNumber = 5
```

```
For StartNumber = 1 To EndNumber

    Cells(StartNumber, "A").Value = StartNumber

Next StartNumber
```

The only difference between this For loop and the last one in the previous section is this line:

```
Cells(StartNumber, "A").Value = StartNumber
```

Because the **StartNumber** gets 1 automatically added to it each time round the loop, we can use it between the round brackets of **Cells**. So the code is doing this:

```
Cells(1, "A").Value = 1
Cells(2, "A").Value = 2
Cells(3, "A").Value = 3
Cells(4, "A").Value = 4
Cells(5, "A").Value = 5
```

Or we could keep the Row number the same and change the column number:

```
Cells(1, StartNumber).Value = StartNumber
```

Or we could do both:

```
Cells(StartNumber, StartNumber).Value = StartNumber
```

The above code will get us a diagonal line form cell A1 to cell E5, which each cell filled with the numbers 1 to 5.

The point is, though, that we can manipulate the cells on a spreadsheet by using just a number from our loop and the Cells property.

You can use **Offset** with cells, too. In this next exercise, we'll do the 10 times table. It will look like this:

	A	B
1	1 times 10 =	10
2	2 times 10 =	20
3	3 times 10 =	30
4	4 times 10 =	40
5	5 times 10 =	50
6	6 times 10 =	60
7	7 times 10 =	70
8	8 times 10 =	80
9	9 times 10 =	90
10	10 times 10 =	100

Create a new Sub and call it **TimesTable**. Set up the following four Integer variables, first:

```
Dim StartNumber As Integer
Dim EndNumber As Integer
Dim answer As Integer
Dim TimesTable As Integer
```

Now put a value of 10 in two of them:

```
EndNumber = 10
TimesTable = 10
```

This time, our end number is 10. We've also specified 10 for the **TimesTable** variable.
The first line of the **For** loop is as before:

```
For StartNumber = 1 To EndNumber
```

And so is the end of the loop:

```
Next StartNumber
```

In between **For** and **Next** the first line is this:

```
answer = StartNumber * TimesTable
```

To work out what this does, start on the right of the equal sign:

```
StartNumber * TimesTable
```

The TimesTable variable will always be 10. But the StartNumber variable changes each time round the loop. The first time round the loop we'll have this:

```
1 * 10
```

The second time round the loop we'll have this:

```
2 * 10
```

The third time it's:

```
3 * 10
```

And so on, right up to 10 times 10.

Each time VBA works out the result of the multiplication, it stores it in the variable called **answer**.
The next line of the code is this:

```
Cells(StartNumber, 1).Value = StartNumber & " times " & TimesTable
& " = "
```

It's rather a long line, so let's break it down to see what's happening.

To the right of the equal sign, we have this:

```
StartNumber & " times " & TimesTable & " = "
```

We're using three concatenation symbols, here (&). So we're joining four things together:

StartNumber
" times "
TimesTable
" = "

We're joining together: the **StartNumber** variable, the direct text " times ", the **TimesTable** variable, and the direct text " = ".
Once VBA has stitched all this together it will place the result into whatever is on the left of the equal sign. Which is this:

```
Cells(StartNumber, 1).Value
```

So the result of the concatenation is being stored in the **Value** property of **Cells**. The cells we're accessing will change because of what we have between the round brackets:

```
StartNumber, 1
```

StartNumber will change each time round the loop, but the hard-coded 1 means the "A" column. So it will be this, the first few times round:

```
Cells(1, "A")
Cells(2, "A")
Cells(3, "A")
Cells(4, "A")
```

The third and final line to add to your **For** loop is this:

Cells(StartNumber, 1).Offset(, 1).Value = answer

The Offset moves the column over 1 from where we were, which was the A column. Whatever is in the answer variable is what will be used as the Value for the cells being referred to with Offset.

The whole of your code should now look like this:

```
Sub TimesTable()

    Dim StartNumber As Integer
    Dim EndNumber As Integer
    Dim answer As Integer
    Dim TimesTable As Integer

    EndNumber = 10
    TimesTable = 10

    For StartNumber = 1 To EndNumber

        answer = StartNumber * TimesTable
        Cells(StartNumber, 1).Value = StartNumber & " times " & TimesTable & " = "
        Cells(StartNumber, 1).Offset(, 1).Value = answer

    Next StartNumber

End Sub
```

Try it out. When your run the code you should see the same values as ours (We've formatted the cells slightly):

	A	B
1	1 times 10 =	10
2	2 times 10 =	20
3	3 times 10 =	30
4	4 times 10 =	40
5	5 times 10 =	50
6	6 times 10 =	60
7	7 times 10 =	70
8	8 times 10 =	80
9	9 times 10 =	90
10	10 times 10 =	100

And there you go - the 10 times table using For loops, Cells, and Offset. But before we move on, try these exercises.

Exercise
Make one single change to your code to display the 12 times table up to a value of 120.
Exercise
Make another single change to print out the 12 times table up to a value 240.
Source:
http://www.homeandlearn.org/the_cells_property.html