Laboratorio Economia e Finanza

uniss
UNIVERSITÀ DEGLI STUDI DI SASSARI

Marco Delogu

Loops in Visual Basic for Excel

Recap and Outline of the Lecture

- Review some basic feature of Excel, Learnt some key Excel functions
- Studied properties of the Poisson Function to build a Poisson Prediction Model. Wrote VBA routines.
- Here More about Loops in Visual Basic. What we learn here can easily be extended to any other programming language
- The topic of this lecture in itself would require much more time. However You are going to learn some tricks that will make you able to use Excel more efficiently
- We consider the syntax of some loop structures and solve right away some simple task
- You can complement this material with **A lot** of resources available in internet. Part of this lecture is adapted link

Exercise (1)

- Consider the excel file Exercise. Consider the excel sheet exercise
- Write a VBA macro which reports in cell (1,B) of the sheet exercise the sum of the values contained in column A
- Change the code to sum the values of column A only if the value in the cell is equal to 2
- Hints:
  - Look at the outline lecture..at the last line of the code where I gave you an example of a VBA routine, method to to determine the last cell that contains data
  - Notice that we have already learnt how to use the if then else construct

Exercise (2)

The column is the one that you find in excel workbook Exercise1. First we want to write vba routine that reports in cell(1,B) the sum of all the values in column A. However it is much better to write one that works for any other similar excel sheet. Notice that I put a value also in cell (322,A).

2
3
2
2
3
2
2

For Next Loop

The ‘For Next’ loop allows you to go through a block of code for the specified number of times.

```
For Counter = Start To End [Step Value]
Code Block to Execute
Next [counter]
```

First we make just attempt that does not actually solve the exercise. We write a code that sum up values stored in Column *A* till cell 12.

The Not perfect solution

Notice that the code below does not solve the exercise.. we are stopping at cell (12,A). However, step by step, we are getting closer to get the full solution

```
Sub sol1()
foglio = "exercise"
endvalue = 12
For i = 1 To endvalue
Worksheets(foglio).Cells(1, 2) = Worksheets(foglio).Cells(1, 2)
+ Worksheets(foglio).Cells(i, 1)
Next i
End Sub
```

Explanations

$$foglio = \text{"}exercise\text{"}$$

The line of code above saves in the variable *foglio* the name of the worksheet on which we want work on.

$$i = 1$$

is our Counter and starts equal to 1

$$endvalue = 12$$

tells us how many times the code inside of the loop will be repeated. Also, the step value in our simple case is equal to 1!

$$Worksheets(foglio).Cells(1,\ 2) = Worksheets(foglio).Cells(1,\ 2) \\ + Worksheets(foglio).Cells(i,\ 1)$$

This line of code adds to cell(1,2) the value of the previous sum..loop!!

Full Solution (1)

VBA is full of already built in methods that we can exploit to improve our code. For instance we can easily exploit the lines of code reported below to determine the last cell, in a given column (in our case column *A*, that contains data as follows:

```
endvalue = Worksheets(foglio).Cells(Rows.Count,
    "A").End(xlUp).Row
```

Full Solution 2

```
Sub sol2()
foglio = ”exercise”
endvalue = Worksheets(foglio).Cells(Rows.Count,
”A”).End(xlUp).Row
For i = 1 To endvalue
Worksheets(foglio).Cells(1, 3) = Worksheets(foglio).Cells(1, 3)
+ Worksheets(foglio).Cells(i, 1) Next i
End Sub
```

Recap

- Notice that our last solution works with minor changes for similar problems. Our first attempt fits only the specific case where the last cell with data is actually cell (12,A)!
- To build the final *solution* we take advantage of some methods that excel objects have already built in
- To deal with the last point we need to include the *if then else*construct into our VBA routine.
- Notice that I am storing the result of the full solution in cell (1,C) to allow you to see the difference

Last point Change the code to sum the values of column A only if the value in the cell is equal to 2. Namely we loop through the values but we keep adding only if the value of the cell equals 2!

Solution

```
Sub sol3()
foglio = "exercise"
endvalue = Worksheets(foglio).Cells(Rows.Count,
"A").End(xlUp).Row
For i = 1 To endvalue
If (Worksheets(foglio).Cells(i, 1) = 2) Then
Worksheets(foglio).Cells(1, 4) = Worksheets(foglio).Cells(1, 4)
+ Worksheets(foglio).Cells(i, 1)
Else
Worksheets(foglio).Cells(1, 4) = Worksheets(foglio).Cells(1, 4)
End If
Next i
End Sub
```

Step

Notice that in our previous code we did not explicitly define the **Size** of the step. We can define it with a minor change into the code. Let assume that we want to write a vba routine that gives the sum of the first 5 even number (2,4,6,8,10). You find the solution in the excel sheet example2 and in the VBA editor, routine labeled sumeven. I report the result into the cell (1,a) of the worksheet example2

Code

```
Sub sumeven()
foglio = ”example2”
For Count = 2 To 10 Step 2
Total = Total + Count
Next Count
Worksheets(foglio).Cells(1, 1) = Total
End Sub
```

Do While Loop

A 'Do While' loop allows you to check for a condition and run the loop while that condition is met (or is TRUE). There are two types of syntax in the Do While Loop.

```
Do [While condition]
Code block to Execute
Loop
```

OR

```
Do
Code block to Execute
Loop [While condition]
```

Do While Loop

The difference between the two is that in the first, the While condition is checked before any code block is executed. On the contrary, in the second case, the code block is executed first and then the While condition is checked.
This means that if the While condition is False, the code will still run at least once in the second case (as the ‘While’ condition is checked after the code has been executed once).
Now let’s see some examples employing the *Do While loops* in VBA.

Do While: Example

Suppose you want to add the first ten positive integers using the Do While loop in VBA.
To solve this task, you can use the Do While loop until the next number is less than or equal to 10. As soon as the number is greater than 10, your loop should stop. We save the output of the routine in the excel sheet dowhile, cell (1,1).
We can solve the problem inserting the while condition either at the beginning or at the end of the loop

Solution (1)

```
Sub dowhile1()
foglio = "dowhile1"
i = 1
Do While i <= 10
result = result + i
i = i + 1
Loop
Worksheets(foglio).Cells(1, 1) = result
End Sub
```

Solution (2)

```
Sub dowhile2()
foglio = ”dowhile2”
i = 1
Do
result = result + i
i = i + 1
Loop While (i <= 10) Worksheets(foglio).Cells(1, 1) = result
End Sub
```

Do Until Loops

Do Until' loops are very much like the 'Do While' loops.
In 'Do While', the loop runs till the given condition is met, while in 'Do Until', it loops until the **specified condition is met**.
There are two types of syntax in the Do Until Loop.

Do Until Loops (2)

```
Do [Until condition]
Code block to Execute
Loop
```

```
Do
Code block to Execute
Loop [Until condition]
```

Similarly to the DoWhile construct in the second case the condition is checked after that the code is run at least one time

Solution (1)

We solve taking advantage of the Do Loop Until construct the same problem that we solved with the Do Until construct.

```
Sub dountil1()
foglio = "dountil1"
i = 1
Do Until (i > 10)
result = result + i
i = i + 1
Loop
Worksheets(foglio).Cells(1, 1) = result
End Sub
```

Solution (2)

```
Sub dountil2()
foglio = "dountil2"
i = 1
Do
result = result + i
i = i + 1
Loop Until (i > 10)
Worksheets(foglio).Cells(1, 1) = result
End Sub
```

Exercises

1. Write a Visual Basic routine that gives you the product of the first 10 integer
2. Write a vba routine that computes the product of the numbers between 20 and 25. However if the number is larger than 1000 it must write ”too large”
3. Make a research in internet and look for the method clear content. Consider exercise 1. write a code that deletes all numbers different from 2 from cell 2. I am a nice guy, you got the method in the file
4. Write a routine that reports in the column D of the excel sheet the numbers reported in column A if and only if they are lower than the number reported in cell $(1, C$. Use either *while* or *until*. Notice that the code must be flexible!