Visual Basic New
Yet Another Kind of Loop
Isn’t one way enough

Loop Basics
- Up to this point we have looked at two types of loops
- The For_Next loop

```
For counter_variable = starting_value to limiting_value step increment_value
    loop actions
Next counter_variable
```

Loop Basics
- And the For Each_Next loop

```
For Each collection_object In collection
    Loop Actions
Next collection_object
```

Loop Basics
- The For_Next loop is used to complete some action or actions a number of times based on counting from the starting_value to the ending_value in steps of the step_value.
The For Each...Next loop works on all the members of a collection performing the actions within the loop on every member of the collection.

There is a third type of loop that works based on a logical test that is made either before or after the loop executes.

This is a subtle point and one that can be easily missed.

One of these new types of loops is called a post-test loop, the other is known as a pre-test loop.

All of these loops are collectively known as Do loops.

Even though they vary in form a bit, all have the characteristic that they are based on a logical condition.

Rather than count or work on all the members of a collection, we will check a logical condition each time we work on the loop.

The first form we will consider is the Do While...Loop.

```
Do While logical_expression
  loop actions
Loop
```

The flowchart form of this type of loop is:

**Do While** logical_expression
loop actions
Loop

Is the logical expression true?

Yes

No

As long as the logical expression evaluates to true, the loop will continue to execute.

Some statement inside of the loop must change a value so that the logical expression eventually evaluates to false.

If not, then the loop will continue to run as long as the computer is running the program.
Do Loop

- This is a very long time unless you turn off the program and the machine.

Do While logical_expression
  loop actions
Loop

Do Loop

- Here is a very simple example of this type of loop.

Do Loop

- We start out by setting Total to 0.
- This isn’t a loop control variable as in the For_Next loops.

Do Loop

- When we hit the loop for the first time, we evaluate the logical expression.
Do Loop
- Since 0 is less than 20, it evaluates true and we execute the statements inside the loop.

Do Loop
- The statement inside the loop adds 1 to Total.

Do Loop
- What number would you expect to be in cell A1 after this program executes?
  19 or 20?
Do Loop

- The loop will only stop executing when the condition evaluates to false.
- The evaluation is only false if Total is not less than 20.

Do Loop

- At Total equal to 19, the evaluation is true so the loop continues.
- At Total equal to 20, the evaluation is false so the loop terminates.

Do Loop

- Now we are going to change the program slightly and see if we can predict what the program will do.
- Rather than adding 1 each time through the loop we are adding 3.

Do Loop

- We can make a table of events and values and see what is happening in the loop.
Do Loop

- We will start at the first entry to the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>true</td>
</tr>
</tbody>
</table>

Do Loop

- Since the logical condition is true we make a pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>true</td>
</tr>
</tbody>
</table>

Do Loop

- Since the logical condition is still true we make another pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>true</td>
</tr>
</tbody>
</table>

Do Loop

- Since the logical condition is still true we make another pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>true</td>
</tr>
</tbody>
</table>
Do Loop

• Since the logical condition is still true we make another pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>true</td>
</tr>
</tbody>
</table>

Do Loop

• Since the logical condition is still true we make another pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>true</td>
</tr>
</tbody>
</table>

Do Loop

• Since the logical condition is still true we make another pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>true</td>
</tr>
</tbody>
</table>

Do Loop

• Since the logical condition is still true we make another pass through the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>true</td>
</tr>
</tbody>
</table>
Do Loop

- Now at the top of the loop the condition evaluates to false and we exit the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>false</td>
</tr>
</tbody>
</table>

Do Loop

- When this happens, we exit the loop

<table>
<thead>
<tr>
<th>Total</th>
<th>Total&lt;20?</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>false</td>
</tr>
</tbody>
</table>

Do Loop

- The order in this type of loop is:
  - Test the logical condition
  - If it evaluates to true
    - Execute the statements inside the loop
  - If it evaluates to false
    - Exit the loop

Do Loop

- There must be something within the loop that will eventually change the logical evaluation from true to false

Public Sub DoWhileLoop()
  Dim Total As Single
  Total = 0
  Do While Total < 20
    Total = Total + 2
    DoSub
  Loop
  MsgBox("Total = " & Total)  ' Exit the loop
End Sub

Public Sub DoWhileLoop()
  Dim Total As Single
  Total = 0
  Do While Total < 20
    Total = Total + 2
    DoSub
  Loop
  MsgBox("Total = " & Total)  ' Exit the loop
End Sub

Public Sub DoWhileLoop()
  Dim Total As Single
  Total = 0
  Do While Total < 20
    Total = Total + 2
    DoSub
  Loop
  MsgBox("Total = " & Total)  ' Exit the loop
End Sub

Public Sub DoWhileLoop()
  Dim Total As Single
  Total = 0
  Do While Total < 20
    Total = Total + 2
    DoSub
  Loop
  MsgBox("Total = " & Total)  ' Exit the loop
End Sub
Do Loop

- If the logical condition evaluates to false before the first execution of the loop then the loop never evaluates.

Do Loop

- In the loop at the right, the loop will never execute because Total is already >= 20 before we enter the loop.

Do Loop

- A slightly different form of a pretest loop is the Do Until....Loop form.
- In this case, we will execute the loop until the logical condition becomes true.

Do Loop

- Notice that the Yes and No branches have reversed places.
- Now the loop will continue to execute until the logical condition becomes true.
We can change the code from our previous example to utilize this type of loop.

```vbnet
Public Sub DoUntilLoops()
    Dim Total As Single
    Total = 0
    Do Until Total >= 20
        Total = Total + 3
        Loop
    Range("A1").Value = Total
End Sub
```

Notice that we have changed our logical expression from `Total < 20` to `Total >= 20`.

This is because we want to execute the loop until this is achieved.

We can run the programs side by side and see how each operates.

<table>
<thead>
<tr>
<th>Do While</th>
<th>Do Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Total &gt;= 20</td>
</tr>
<tr>
<td>Total + 3</td>
<td>Total + 3</td>
</tr>
</tbody>
</table>

We can run the programs side by side and see how each operates.
Do Loop – Side By Side Comparison

We start each program and go to the point where we give a value to Total.

Do While Total < 20
Total = Total + 3
Loop
Range("Ai").Value = Total
End Sub

Do Until Total >= 20
Total = 0#
Do Until Total >= 20
Total = Total + 3
Loop
Range("Ai").Value = Total
End Sub

Now we move to the top of the loop for the first time and make the test.

Do While Total < 20
Total = Total + 3
Loop
Range("Ai").Value = Total
End Sub

When we go back to the top of each loop, we check the logical conditions again.
Again, we execute.

Do While Total < 20
Total = Total + 3
Loop
Range("Ai").Value = Total
End Sub

Do Until Total >= 20
Total = 0#
Do Until Total >= 20
Total = Total + 3
Loop
Range("Ai").Value = Total
End Sub

When we go back to the top of each loop, we check the logical conditions again.
Again, we execute.
Do Loop – Side By Side Comparison

When we go back to the top of each loop, we check the logical conditions again. Again, we execute.

<table>
<thead>
<tr>
<th>Do While</th>
<th>Do Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Total=20?</td>
</tr>
<tr>
<td>12</td>
<td>True</td>
</tr>
</tbody>
</table>

Do Loop – Side By Side Comparison

When we go back to the top of each loop, we check the logical conditions again. Again, we execute.

<table>
<thead>
<tr>
<th>Do While</th>
<th>Do Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Total=20?</td>
</tr>
<tr>
<td>15</td>
<td>True</td>
</tr>
</tbody>
</table>

Do Loop – Side By Side Comparison

When we go back to the top of each loop, we check the logical conditions again. Again, we execute.

<table>
<thead>
<tr>
<th>Do While</th>
<th>Do Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Total=20?</td>
</tr>
<tr>
<td>18</td>
<td>True</td>
</tr>
</tbody>
</table>

Do Loop – Side By Side Comparison

When we go back to the top of each loop, we check the logical conditions again. Again, we execute.

<table>
<thead>
<tr>
<th>Do While</th>
<th>Do Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Total=20?</td>
</tr>
<tr>
<td>21</td>
<td>True</td>
</tr>
</tbody>
</table>
Do Loop – Side By Side Comparison

Public Sub DoWhileLoop()
  Dim Total As Single
  Total = 0# 
  Do While Total < 20
    Total = Total + 3
    Range("A1").Value = Total 
  Loop
End Sub

Public Sub DoUntilLoop()
  Dim Total As Single
  Total = 0# 
  Do Until Total >= 20
    Total = Total + 3
    Range("A1").Value = Total 
  Loop
End Sub

Now when we get back to the top of each loop, both logical expressions change value.

<table>
<thead>
<tr>
<th>Do While</th>
<th>Do Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total 20?</td>
<td>Total &gt;= 20?</td>
</tr>
<tr>
<td>21</td>
<td>False</td>
</tr>
</tbody>
</table>

The exit conditions for both loops are satisfied.

You might be asking who have two ways to do essentially the same thing. The main difference is in how you want to frame the exit conditions from the loop. If you want to exit when a condition is true, use a Do Until...Loop. If you want to exit when a condition is false, use a Do While...Loop.

Old Program Revisited-

- We looked at a way to find the balance elevation by submitting better guesses as we executed the program.
- We took advantage of using a loop (FOR_NEXT...LOOP) to do all the cells within a selection.
- Now we can expand this just a bit.
Consider this code

There are a number of new features in this code. Look over the code and determine if you can tell what it is doing.

Homework

- In 1112, you used the bisection method to solve for the root of a function.
- That method can also be used to make this program much more efficient.
- For Monday the 2nd, you should submit a flow chart for a program to do what this program does but uses the Method of Bisection rather than how we have done this today.

This is a formal submission.
- You can review the details of the Bisection method on Dr. Camp's 1112 web site at www.ce.memphis.edu/1112
New Type of Loop

And here is a run with a really great guess