Loop Basics

- We have looked at the For_Next loop in the previous class
- The form of the structure was

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

Loop Basics

- Usually we integer values for the counter_variable which means that the starting_value and limiting_value must be either a integer constant or an integer variable

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

Public Sub LoopExamples()
    Dim FirstValue As Integer: FirstValue = Range("A1").Value
    Dim LastValue As Integer: LastValue = Range("A10000").Value
    Dim OffSet As Integer
    Dim Value
    OffSet = 1
    Do While FirstValue <= LastValue
        Value = FirstValue + OffSet
        Range("A1").Select
        Range("A1").Value = Value
        Range("A2").Select
        Range("A2").Value = Value
        OffSet = OffSet + 1
        LoopNext
       End Do
End Sub

New Type of Loop

- Now we will look at another type of loop that is very convenient when working with spreadsheets
- It allows us to work with groups of selected cells that only have to be defined by being selected by the user

In some of our previous programs, we selected specific cells by hardwiring them into our program

Hardwiring means that we defined them internally to the program and did not give the user of the program the ability to change their location. To change them would require that the program itself is changed

We use the syntax

Range("A1").Select

Where A1 refers to the cell at location A1 on the spreadsheet
We can also select a specific group of cells by using an alternate syntax:

```
Range("A1:C3").Select
```

Now we select all the cells bounded by a box with cell A1 in the upper left to cell C3 in the lower right.

In each of these cases, we specified exactly what cells we wanted to work on or with and hardwired their location in the spreadsheet.

What if we want to work with a group of cells that are located anywhere on the spreadsheet?
New Type of Loop

The EXCEL object model has what are known as collections
A collection is a group containing like objects
We never specifically state that something is a collection we just take advantage the fact that collections exist

Since every cell in a spreadsheet is actually a range object (as well as other types of objects) we can gather the cells we want to utilize into a collection and then use a special form of a loop to work on all the cells
We did this in 2107 but without any explanation

The first step in utilizing this is to set up a variable that has as its type, the object type rather than the numeric type
We will limit ourselves for the moment to using the Range type so our syntax will be

We will use the Selection object to hold all the individual range objects
This is much easier than it looks
All we have to do if just select a range of cells with the cursor on the spreadsheet
We have started in cell B4 and held down the mouse button until we reached cell P10. All the shaded cells are now available as the selection object.

The highlighted cells are in the collection within the Selection object.

We can utilize a new type of loop to work on all the cells within that selection object.

Dim variable_name as Range

The syntax of the loop we will use is

For Each collection_object In collection
    Loop Actions
Next collection_object
New Type of Loop

• For example
  - We have a spreadsheet with a set of elevations stored in a group of cells
  - We don't know how many before we write the program, actually we don't even need to know what shape the cells form
  - We want to set some elevation so that there is as much elevation above that elevation as there is below

New Type of Loop

• We will start with our usual beginning setting up a description of the program as well as forcing all variables to be typed

New Type of Loop

• We can all the user to input their guess as to the elevation which will balance the system in cell A1 and we will store that value in a variable named BaseGuess

New Type of Loop

• As we work through the cells, we need to keep a running total of the differences between the base elevation guess (BaseGuess) and the elevation in each cell
• We will set up another variable to hold this
New Type of Loop

- We will name this variable Volume

In order to use our new loop structure, we need to set up a range object that will be used to temporarily hold all the range objects within our collection.

We will label this temporary object testCell

We will get the balancing elevation guess from cell A1 and store it in BaseGuess
New Type of Loop

- Now we need to initialize the volume since it is a running total.

**Example Code**

```vba
Function InitializeVolume() _
    Dim Range As Range, Volume As Double
    Set Range = Selection
    Volume = 0
End Function
```

Now we get to the loop structure.

- We are going to be using `testCell` to hold each member of the collection of range objects in the selection object as we operate on it.
- This is the reason why we typed `testCell` as `Range` rather than the types we had used in the past.

Remember that the selection object automatically includes all the cells that we have selected by highlighting them in the worksheet.

We set the loop up first without anything inside of the loop.
What we have said is that one at a time, for every range object within the collection of range objects within selection, perform what is within the loop.

At the moment it is nothing.

Since we want to find the balancing elevation, we will find the difference between each elevation and sum these differences.

The code to do this is:

```
For Each testCell In Selection
    Volume = Volume + testCell.Value - BestGuess
Next testCell
```

This reads:

- For each range object within the selection
  - Store the range object in the range object testCell
  - Subtract the value in BestGuess from the value property of the current value of the range object testCell
  - Add this to the current value of Volume
  - Store this in Volume
New Type of Loop

Finally we will put the value of volume into a cell for the user to see.

We can look at an example of how this program works.
We will start with a group of cells.

We will then select out guess for the base elevation and put it into A1.
Now we select the range of cells we want to work on.

These will be the ones that we include in our selection object and therefore the collection of range objects.

The cells B2 through K29 are included.

Now we can run our macro and see how close our guess is.
Since we have a positive number, our guess for the balance elevation must have been too low.

We could continue to make guesses and further refine the solution by running the macro multiple times with multiple guesses.

Just as an additional step, we can color code the cells based on above and below our balanced elevation guess.

To do this, all we need to do is add some conditional information within the loop.

Here is an example of that code:

```plaintext
Public Sub ExampleSub()
    Dim testCell As Range
    Dim Volume As Double
    Dim testCellValue As Double

    testCell = Range("A1")
    Volume = 0

    For Each testCell In Range(testCell, Range("A10000"))
        testCellValue = testCell.Value
        If testCellValue > Volume Then
            testCell.Font.Color = Red
        ElseIf testCellValue < Volume Then
            testCell.Font.Color = Blue
        Else
            testCell.Font.Color = Black
        End If
    Next testCell
End Sub
```
New Type of Loop

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>1.5</td>
<td>1.6</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
</tr>
<tr>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
<td>2.4</td>
<td>2.5</td>
<td>2.6</td>
<td>2.7</td>
<td>2.8</td>
<td>2.9</td>
<td>3.0</td>
<td>3.1</td>
</tr>
<tr>
<td>3.1</td>
<td>3.2</td>
<td>3.3</td>
<td>3.4</td>
<td>3.5</td>
<td>3.6</td>
<td>3.7</td>
<td>3.8</td>
<td>3.9</td>
<td>4.0</td>
<td>4.1</td>
</tr>
<tr>
<td>4.1</td>
<td>4.2</td>
<td>4.3</td>
<td>4.4</td>
<td>4.5</td>
<td>4.6</td>
<td>4.7</td>
<td>4.8</td>
<td>4.9</td>
<td>5.0</td>
<td>5.1</td>
</tr>
<tr>
<td>5.1</td>
<td>5.2</td>
<td>5.3</td>
<td>5.4</td>
<td>5.5</td>
<td>5.6</td>
<td>5.7</td>
<td>5.8</td>
<td>5.9</td>
<td>6.0</td>
<td>6.1</td>
</tr>
<tr>
<td>6.1</td>
<td>6.2</td>
<td>6.3</td>
<td>6.4</td>
<td>6.5</td>
<td>6.6</td>
<td>6.7</td>
<td>6.8</td>
<td>6.9</td>
<td>7.0</td>
<td>7.1</td>
</tr>
<tr>
<td>7.1</td>
<td>7.2</td>
<td>7.3</td>
<td>7.4</td>
<td>7.5</td>
<td>7.6</td>
<td>7.7</td>
<td>7.8</td>
<td>7.9</td>
<td>8.0</td>
<td>8.1</td>
</tr>
<tr>
<td>8.1</td>
<td>8.2</td>
<td>8.3</td>
<td>8.4</td>
<td>8.5</td>
<td>8.6</td>
<td>8.7</td>
<td>8.8</td>
<td>8.9</td>
<td>9.0</td>
<td>9.1</td>
</tr>
</tbody>
</table>

And here is a run with a really great guess.