overview of values and formulas

Values and formulas will focus on what to do with a Range after you have it. This typically falls into a couple of categories:

- Do some control logic based on the value of the Cell
- Apply some formatting to the cell
- Modify the formulas of the cell
- Manipulate the cell or its neighbors in order to produce a more useful result
- Transform the cells based on their content
- Do something specific to Excel with the `Range`: conditional formatting, data validation, comment, hyperlink etc.
- adding or deleting a `Range` or possibly just using one of the clear function

In addition to those basic tasks, also include:

- Working with Conditional Formatting
- Combining some more advanced topics like using the data in a `Range` to manipulate something about a `Chart`

TODO: run through bUTL and see what other category of things there are

chapter 2 - 1, introduction to manipulations

This chapter will focus on the actual work of using `Ranges` for some purpose. This chapter is predicated on the previous one which focused on obtained a `Range`. When talking of "using" a `Range`, the goal usually takes one of the following forms.

Work through a spreadsheet of data, processing it from one format into another. This can be to pull data out, do calculations on a subset of data, change the formatting, aggregate data, summarize data into a new spreadsheet, etc. The options here are really endless, but the main idea is that you have an existing spreadsheet of data to do something with.

The next category of work is to process some small amount of data in place, typically to clean up data or convert it to some other form. A lot of this type of work is providing some functionality that would be great to have in Excel by default. This work also includes a lot of very specific types of functions that only make sense with your data. In that sense, these types of actions can be the quickest hitters; they are specific to your task and easy to program.

Another category of work is to run through an existing worksheet and perform some amount of checking on it. These checks do not necessarily need to modify the spreadsheet, they can be checking for formula errors, bad values, etc.

Another type of work that can be done is to modify the spreadsheet to make it easier to do work or to manage a workflow. These types of things are often implemented as events, but they can just be stray macros as well. When modifying the spreadsheet, you are often showing/hiding columns or worksheets. You can also be sorting `Worksheets`. You might be moving some number of `Worksheets` over from a "template" and setting up a common work environment.

As we progress down this list, things are becoming increasingly complicated. At some point, the work involved will progress from a couple of simple tasks to a much more involved workflow. It's generally the nature of a complicated workflow that it is simply doing a long string of simple tasks in series. In that sense, if you can learn these techniques, you can start to become comfortable combining them in more complicated fashions.

simple manipulations (one steppers)

This section will focus on simple manipulations. Simple manipulations generally take a two step process:

1. Identify a `Range` to work with
2. Do something to that `Range`

In a lot of cases, the `Range` contains multiple cells and may be iterated through a cell at a time to apply the

action.

These simple manipulations truly are simple. It includes things like:

- Change the value of a cell
- Change the value of a group of cells
- Change the formula of a cell
- Change the formatting of a cell
- Clear the formatting or value from a cell
- Return some piece of information about a cell

TODO: deal with these later

Name a group of cells

Add a hyperlink to the current cell

TODO: add a couple examples of the simple manipulations

slightly more complicated manipulations (the two steppers)

This section will on the so called “two steppers”. I call them that because these manipulations typically involve two commands after identifying a `Range`. the first command is usually a logic or loop, and the

second command is the actual work to be done. Two steppers are important because a large number of complicated tasks involve nesting and combining these two steps.

Some examples of two step manipulations includes;

- Run through a list of cells, if the text is numeric, convert to a number
- Run through a list of cells, if the cell is blank, fill with the value from above
- Run through some cells, check if the row is odd or even, and color the row from one of two colors
- Run through one list of cells, apply the formatting to the same cell in a different column

TODO: find some better examples for these as well

strategy #1, do something if

This strategy really is the core of all advanced VBA development. It's simple enough: "do something, if". The endless possibilities come from the choices for "do something" and the things that could be checked in the "if". There are a handful of common scenarios that are best covered by storing some utility code (e.g. convert to a number if numeric). Most of these two step solutions though are specific to the task at hand.

In this section, the goal is to show the general form of this strategy with a couple of examples.

TODO: add a couple examples of this

strategy #2, work through one Range and apply to another Range

This strategy comes up frequently when working through `Ranges` that are related somehow. The general idea is that you want to apply an action in one `Range` based on something about another `Range`. The simplest case of this is to move a value from `Range` to another. This simple case sometimes reduces to not much more than copying and pasting. Having said that, once you get past the simplest version of it, you will be doing something that copy and paste cannot handle.

TODO: add a couple examples of this

TODO: look for more examples of two-step processes

things to change and check

This section will focus on the common properties that are checked and changed with these types of manipulations.

properties of the Range

The common properties of the `Range` to work with include:

- `Value`
- `Text`
- `Formula`
- `Font`
- `Interior`
- `NumberFormat`

TODO: add some examples of working with these

commonly used VBA functions

In addition to the properties of the `Range`, there are a handful of common VBA functions that come up when working with simple to moderate manipulations. These include:

- `Split` - split a string into an array based on a delimiter (the reverse of Join)
- `Join` - join an array into a string with a delimiter (this reverse of Split)
- `Asc` - determine the ASCII code for a character
- `Chr` - return a character for an ASCII code (the reverse of Asc)
- `InStr` - determine if a string is in another one (called Substring in other languages)
- `Left`, Mid, Right - grab parts of a string

- `Trim` - remove any whitespace from the start or end of a string
- `Len` - determine the length of a string
- `UCase`, LCase - used to force a string to upper or lower case
- `UBound`, LBound - determine the bounds of any array
- `WorksheetFunction` - get access to any Excel functions in VBA
- `IsNumeric`, IsEmpty - check if a number – TODO: add the others here
- `CDbl`, CLng, CBool, CDate - convert a value of one type to another – TODO: add any others
- `Replace` - replace one string in another
- `Application`.Index, Application.Match - these are the VBA versions of the Excel functions
- `Application`.Transpose - convert a 1D array from vertical to horizontal and back
- `Is` Nothing - check if a reference has been set
- `TypeName` - check the type of an object (useful if working with `Variant`)
- `RGB` - useful way to build colors from known red, green, and blue values
- `Count` - common way to get the size of a group, used often to resize an input/output or to check logic

TODO: search through bUTL for other common functions

Colorize.md

The Sub below makes it possible to applied a banded row coloring based on changing values in the rows. That is, it looks at the value in a given cell and compares it to the cell above. If the value has changed, it applies the next color. If the same, it will apply the same color as the row above. This gives a simple demonstration of how it's possible to create simple or complicated formatting rules with VBA.

TODO: clean up the code below to simplify the process and show only the core bits needed.

```
Public Sub Colorize()


    Dim targetRange As Range
    On Error GoTo errHandler
    Set targetRange = GetInputOrSelection("Select range to color")
    Dim lastRow As Long
    lastRow = targetRange.Rows.Count
    Dim interiorColor As Long
    interiorColor = RGB(200, 200, 200)


    Dim sameColorForLikeValues As VbMsgBoxResult
    sameColorForLikeValues = MsgBox("Do you want to keep duplicate values the
```

```
        same color?", vbYesNo)

    If sameColorForLikeValues = vbNo Then

        Dim i As Long
        For i = 1 To lastRow
            If i Mod 2 = 0 Then
                targetRange.Rows(i).Interior.Color = interiorColor
            Else: targetRange.Rows(i).Interior.ColorIndex = xlNone
            End If
        Next
    End If


    If sameColorForLikeValues = vbYes Then
        Dim flipFlag As Boolean
        For i = 2 To lastRow
            If targetRange.Cells(i, 1) <> targetRange.Cells(i - 1, 1) Then
               flipFlag = Not flipFlag
```

```
            If flipFlag Then
                targetRange.Rows(i).Interior.Color = interiorColor
            Else: targetRange.Rows(i).Interior.ColorIndex = xlNone
            End If
        Next
    End If
    Exit Sub
errHandler:
    MsgBox "No Range Selected!"
End Sub
```

CombineCells.md

```
Public Sub CombineCells()


    'collect all user data up front
    Dim inputRange As Range
    On Error GoTo errHandler
```

```
6       Set inputRange = GetInputOrSelection("Select the range of cells to
            combine")
7
8       Dim delimiter As String
9       delimiter = Application.InputBox("Delimiter:")
10      If delimiter = "" Or delimiter = "False" Then GoTo delimiterError
11
12      Dim outputRange As Range
13      Set outputRange = GetInputOrSelection("Select the output range")
14
15      'Check the size of input and adjust output
16      Dim numberOfColumns As Long
17      numberOfColumns = inputRange.Columns.Count
18
19      Dim numberOfRows As Long
20      numberOfRows = inputRange.Rows.Count
21
22      outputRange = outputRange.Resize(numberOfRows, 1)
23
```

```
    'Read input rows into a single string
    Dim outputString As String
    Dim i As Long
    For i = 1 To numberOfRows
        outputString = vbNullString
        Dim j As Long
        For j = 1 To numberOfColumns
            outputString = outputString & delimiter & inputRange(i, j)
        Next
        'Get rid of the first character (delimiter)
        outputString = Right(outputString, Len(outputString) - 1)
        'Print it!
        outputRange(i, 1) = outputString
    Next
    Exit Sub
delimiterError:
    MsgBox "No Delimiter Selected!"
    Exit Sub
errHandler:
```

```
    MsgBox "No Range Selected!"
End Sub
```

ConvertToNumber.md

```
Public Sub ConvertToNumber()

    Dim targetCell As Range
    Dim targetSelection As Range

    Set targetSelection = Selection

    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationManual

    For Each targetCell In Intersect(targetSelection, ActiveSheet.UsedRange)
        If Not IsEmpty(targetCell.Value) And IsNumeric(targetCell.Value) Then
            targetCell.Value = CDbl(targetCell.Value)
    Next targetCell
```

```
14
15      Application.ScreenUpdating = True
16      Application.Calculation = xlCalculationAutomatic
17
18  End Sub
```

CreateConditionalsForFormatting.md

TODO: find a better location for this code. Somewhere formatting related?

```
1   Public Sub CreateConditionalsForFormatting()
2
3       On Error GoTo errHandler
4       Dim inputRange As Range
5       Set inputRange = GetInputOrSelection("Select the range of cells to
            convert")
6       'add these in as powers of 3, starting at 1 = 10^0
7       Const ARRAY_MARKERS As String = " ,k,M,B,T,Q"
8       Dim arrMarkers As Variant
9       arrMarkers = Split(ARRAY_MARKERS, ",")
```

```
10
11      Dim i As Long
12      For i = UBound(arrMarkers) To 0 Step -1
13
14          With inputRange.FormatConditions.Add(xlCellValue, xlGreaterEqual, 10
                ^ (3 * i))
15              .NumberFormat = "0.0" & Application.WorksheetFunction.Rept(",", i
                    ) & " "" " & arrMarkers(i) & """"
16          End With
17
18      Next
19      Exit Sub
20  errHandler:
21      MsgBox "No Range Selected!"
22  End Sub
```

MakeHyperlinks.md

```
1   Public Sub MakeHyperlinks()
```

```
2
3       '+Changed to inputbox
4       On Error GoTo errHandler
5       Dim targetRange As Range
6       Set targetRange = GetInputOrSelection("Select the range of cells to
            convert to hyperlink")
7
8       'TODO: choose a better variable name
9       Dim targetCell As Range
10      For Each targetCell In targetRange
11          ActiveSheet.Hyperlinks.Add Anchor:=targetCell, Address:=targetCell
12      Next targetCell
13      Exit Sub
14  errHandler:
15      MsgBox "No Range Selected!"
16  End Sub
```

SelectedToValue.md

```
Public Sub SelectedToValue()


    Dim targetRange As Range
    On Error GoTo errHandler
    Set targetRange = GetInputOrSelection("Select the formulas you'd like to
        convert to static values")

    Dim targetCell As Range
    Dim targetCellValue As String
    For Each targetCell In targetRange
        targetCellValue = targetCell.Value
        targetCell.Clear
        targetCell = targetCellValue
    Next targetCell
    Exit Sub
errHandler:
    MsgBox "No selection made!"
End Sub
```

SplitAndKeep.md

```
Public Sub SplitAndKeep()

    On Error GoTo SplitAndKeep_Error

    Dim rangeToSplit As Range
    Set rangeToSplit = GetInputOrSelection("Select range to split")

    If rangeToSplit Is Nothing Then
        Exit Sub
    End If

    Dim delimiter As Variant
    delimiter = InputBox("What delimiter to split on?")
    'StrPtr is undocumented, perhaps add documentation or change function
    If StrPtr(delimiter) = 0 Then
        Exit Sub
    End If

```

```
19      Dim itemToKeep As Variant
20      'Perhaps inform user to input the sequence number of the item to keep
21      itemToKeep = InputBox("Which item to keep? (This is 0-indexed)")
22
23      If StrPtr(itemToKeep) = 0 Then
24          Exit Sub
25      End If
26
27      Dim targetCell As Range
28      For Each targetCell In Intersect(rangeToSplit, rangeToSplit.Parent.
            UsedRange)
29
30          Dim delimitedCellParts As Variant
31          delimitedCellParts = Split(targetCell, delimiter)
32
33          If UBound(delimitedCellParts) >= itemToKeep Then
34              targetCell.Value = delimitedCellParts(itemToKeep)
35          End If
36
```

```
    Next targetCell

    On Error GoTo 0
    Exit Sub

SplitAndKeep_Error:
    MsgBox "Check that a valid Range is selected and that a number was
        entered for which item to keep."
End Sub
```

SplitIntoColumns.md

```
Public Sub SplitIntoColumns()

    Dim inputRange As Range

    Set inputRange = GetInputOrSelection("Select the range of cells to split"
        )

```

```
7     Dim targetCell As Range
8
9     Dim delimiter As String
10    delimiter = Application.InputBox("What is the delimiter?", , ",",
          vbOKCancel)
11    If delimiter = "" Or delimiter = "False" Then GoTo errHandler
12    For Each targetCell In inputRange
13
14        Dim targetCellParts As Variant
15        targetCellParts = Split(targetCell, delimiter)
16
17        Dim targetPart As Variant
18        For Each targetPart In targetCellParts
19
20            Set targetCell = targetCell.Offset(, 1)
21            targetCell = targetPart
22
23        Next targetPart
24
```

```
    Next targetCell
    Exit Sub
errHandler:
    MsgBox "No Delimiter Defined!"
End Sub
```

SplitIntoRows.md

```
Public Sub SplitIntoRows()

    Dim outputRange As Range

    Dim inputRange As Range
    Set inputRange = Selection

    Set outputRange = GetInputOrSelection("Select the output corner")

    Dim targetPart As Variant
    Dim offsetCounter As Long
```

```
    offsetCounter = 0
    Dim targetCell As Range

    For Each targetCell In inputRange.SpecialCells(xlCellTypeVisible)
        Dim targetParts As Variant
        targetParts = Split(targetCell, vbLf)

        For Each targetPart In targetParts
            outputRange.Offset(offsetCounter) = targetPart

            offsetCounter = offsetCounter + 1
        Next targetPart
    Next targetCell
End Sub
```

TrimSelection.md

```
Public Sub TrimSelection()

```

```
3       Dim rangeToTrim As Range
4       On Error GoTo errHandler
5       Set rangeToTrim = GetInputOrSelection("Select the formulas you'd like to
            convert to static values")
6
7       'disable calcs to speed up
8       Application.ScreenUpdating = False
9       Application.EnableEvents = False
10      Application.Calculation = xlCalculationManual
11
12      'force to only consider used range
13      Set rangeToTrim = Intersect(rangeToTrim, rangeToTrim.Parent.UsedRange)
14
15      Dim targetCell As Range
16      For Each targetCell In rangeToTrim
17
18          'only change if needed
19          Dim temporaryTrimHolder As Variant
20          temporaryTrimHolder = Trim(targetCell.Value)
```

```
21
22          'added support for char 160
23          'TODO add more characters to remove
24          temporaryTrimHolder = Replace(temporaryTrimHolder, chr(160),
                vbNullString)
25
26          If temporaryTrimHolder <> targetCell.Value Then targetCell.Value =
                temporaryTrimHolder
27
28      Next targetCell
29
30      Application.Calculation = xlCalculationAutomatic
31      Application.EnableEvents = True
32      Application.ScreenUpdating = True
33
34      Exit Sub
35  errHandler:
36      MsgBox "No Delimiter Defined!"
37      Application.ScreenUpdating = False
```

```
38      Application.EnableEvents = False
39      Application.Calculation = xlCalculationManual
40  End Sub
```