overview of utility code

This chapter might be a dumping ground for useful code that is worth being able to reference. Some things to include here:

- Code to work through the selection on a Chart
- The RangeEnd function to quick get the end of a Range
- Some string processing code?
- The code to work with split values
- Code to convert a 2D array of values to 1D
- GetOrCreateWorksheet which gives you a valid object regardless of what existed
- CreateNextSheet which increments a name as needed
- Creating a Chart based on the XValues, Values, and Name (and order).
- CopyResize command which is used to replicate Copy/PasteValues without using the Clipboard

ColorInputs.md

```
Public Sub ColorInputs()

    Dim targetCell As Range
    Const FIRST_COLOR_ACCENT As String = "msoThemeColorAccent1"
    Const SECOND_COLOR_ACCENT As String = "msoThemeColorAccent2"
    'This is finding cells that aren't blank, but the description says it
       should be cells with no values..
    For Each targetCell In Selection
        If targetCell.Value <> "" Then
            If targetCell.HasFormula Then
                targetCell.Interior.ThemeColor = FIRST_COLOR_ACCENT
            Else
                targetCell.Interior.ThemeColor = SECOND_COLOR_ACCENT
            End If
        End If
    Next targetCell

End Sub
```

CombineAllSheetsData.md

```
Public Sub CombineAllSheetsData()

    'create the new wktk and sheet
    Dim targetWorkbook As Workbook
    Dim sourceWorkbook As Workbook

    Set sourceWorkbook = ActiveWorkbook
    Set targetWorkbook = Workbooks.Add

    Dim targetWorksheet As Worksheet
    Set targetWorksheet = targetWorkbook.Sheets.Add

    Dim isFirst As Boolean
    isFirst = True

    Dim targetRow As Long
    targetRow = 1

    Dim sourceWorksheet As Worksheet
    For Each sourceWorksheet In sourceWorkbook.Sheets
        If sourceWorksheet.name <> targetWorksheet.name Then

            sourceWorksheet.Unprotect

            'get the headers squared up
            If isFirst Then
                'copy over all headers
                sourceWorksheet.Rows(1).Copy targetWorksheet.Range("A1")
                isFirst = False

            Else
                'search for missing columns
                Dim headerRow As Range
                For Each headerRow In Intersect(sourceWorksheet.Rows(1),
                    sourceWorksheet.UsedRange)

                    'check if it exists
```

```
                    Dim matchingHeader As Variant
                    matchingHeader = Application.Match(headerRow,
                        targetWorksheet.Rows(1), 0)

                    'if not, add to header row
                    If IsError(matchingHeader) Then targetWorksheet.Range("A1
                        ").End(xlToRight).Offset(, 1) = headerRow
                Next headerRow
            End If

            'find the PnPID column for combo
            Dim pIDColumn As Long
            pIDColumn = Application.Match("PnPID", targetWorksheet.Rows(1),
                0)

            'find the PnPID column for data
            Dim pIDData As Long
            pIDData = Application.Match("PnPID", sourceWorksheet.Rows(1), 0)

            'add the data, row by row
            Dim targetCell As Range
            For Each targetCell In sourceWorksheet.UsedRange.SpecialCells(
                xlCellTypeConstants)
                If targetCell.Row > 1 Then

                    'check if the PnPID exists in the combo sheet
                    Dim sourceRow As Variant
                    sourceRow = Application.Match( _
                                sourceWorksheet.Cells(targetCell.Row, pIDData)
                                    , _
                                targetWorksheet.Columns(pIDColumn), _
                                0)

                    'add new row if it did not exist and id number
                    If IsError(sourceRow) Then
                        sourceRow = targetWorksheet.Columns(pIDColumn).Cells(
                            targetWorksheet.Rows.Count, 1).End(xlUp).Offset(1)
                            .Row
```

```
                        targetWorksheet.Cells(sourceRow, pIDColumn) =
                            sourceWorksheet.Cells(targetCell.Row, pIDData)
                    End If

                    'get column
                    Dim columnNumber As Long
                    columnNumber = Application.Match(sourceWorksheet.Cells(1,
                        targetCell.Column), targetWorksheet.Rows(1), 0)

                    'update combo data
                    targetWorksheet.Cells(sourceRow, columnNumber) =
                        targetCell

                End If
            Next targetCell
        End If
    Next sourceWorksheet
End Sub
```

ConvertSelectionToCsv.md

```
Public Sub ConvertSelectionToCsv()

    Dim sourceRange As Range
    Set sourceRange = GetInputOrSelection("Choose range for converting to CSV
        ")

    If sourceRange Is Nothing Then Exit Sub

    Dim outputString As String

    Dim dataRow As Range
    For Each dataRow In sourceRange.Rows

        Dim dataArray As Variant
        dataArray = Application.Transpose(Application.Transpose(dataRow.Rows.
            Value2))

```

```
        'TODO:  improve this to use another Join instead of string concats
        outputString = outputString & Join(dataArray, ",") & vbCrLf

    Next dataRow

    Dim myClipboard As MSForms.DataObject
    Set myClipboard = New MSForms.DataObject

    myClipboard.SetText outputString
    myClipboard.PutInClipboard

End Sub
```

CopyCellAddress.md

```
Public Sub CopyCellAddress()


    'TODO: this need to get a button or a keyboard shortcut for easy use
    Dim myClipboard As MSForms.DataObject
    Set myClipboard = New MSForms.DataObject

    Dim sourceRange As Range
    Set sourceRange = Selection

    myClipboard.SetText sourceRange.Address(True, True, xlA1, True)
    myClipboard.PutInClipboard
End Sub
```

CutPasteTranspose.md

```
Public Sub CutPasteTranspose()


    '########Still Needs to address Issue#23##############
    On Error GoTo errHandler
```

```
    Dim sourceRange As Range
    'TODO #Should use new inputbox function
    Set sourceRange = Selection

    Dim outputRange As Range
    Set outputRange = Application.InputBox("Select output corner", Type:=8)

    Application.ScreenUpdating = False
    Application.EnableEvents = False
    Application.Calculation = xlCalculationManual

    Dim topLeftCell As Range
    Set topLeftCell = sourceRange.Cells(1, 1)

    Dim topRow As Long
    topRow = topLeftCell.Row
    Dim leftColumn As Long
    leftColumn = topLeftCell.Column

    Dim outputRow As Long
    Dim outputColumn As Long
    outputRow = outputRange.Row
    outputColumn = outputRange.Column

    outputRange.Activate

    'Check to not overwrite
    Dim targetCell As Range
    For Each targetCell In sourceRange
        If Not Intersect(sourceRange, Cells(outputRow + targetCell.Column -
           leftColumn, outputColumn + targetCell.Row - topRow)) Is Nothing
           Then
            MsgBox ("Your destination intersects with your data. Exiting.")
            GoTo errHandler
        End If
    Next

    'this can be better
    For Each targetCell In sourceRange
```

```
        targetCell.Cut
        ActiveSheet.Cells(outputRow + targetCell.Column - leftColumn,
            outputColumn + targetCell.Row - topRow).Activate
        ActiveSheet.Paste
    Next targetCell

errHandler:
    Application.CutCopyMode = False
    Application.ScreenUpdating = True
    Application.EnableEvents = True
    Application.Calculation = xlCalculationAutomatic
    Application.Calculate

End Sub
```

FillValueDown.md

```
Public Sub FillValueDown()

    Dim inputRange As Range
    Set inputRange = GetInputOrSelection("Select range for waterfall")

    If inputRange Is Nothing Then Exit Sub

    Dim targetCell As Range
    For Each targetCell In Intersect(inputRange.SpecialCells(xlCellTypeBlanks
        ), inputRange.Parent.UsedRange)
        targetCell = targetCell.End(xlUp)
    Next targetCell

End Sub
```

ForceRecalc.md

```
Public Sub ForceRecalc()

```

```
    Application.CalculateFullRebuild

End Sub
```

GenerateRandomData.md

```
Public Sub GenerateRandomData()

    Const NUMBER_OF_ROWS As Long = 10
    Const NUMBER_OF_COLUMNS As Long = 3 '0 index
    Const DEFAULT_COLUMN_WIDTH As Long = 15

    'Since we only work with offset, targetcell can be a constant, but range
       constants are awkward
    Dim targetCell As Range
    Set targetCell = Range("B2")

    Dim i As Long

    For i = 0 To NUMBER_OF_COLUMNS
        targetCell.Offset(, i) = chr(65 + i)

        With targetCell.Offset(1, i).Resize(NUMBER_OF_ROWS)
            Select Case i
            Case 0
                .Formula = "=TODAY()+ROW()"
            Case Else
                .Formula = "=RANDBETWEEN(1,100)"
            End Select

            .Value = .Value
        End With
    Next i

    ActiveSheet.UsedRange.Columns.ColumnWidth = DEFAULT_COLUMN_WIDTH

End Sub
```

OpenContainingFolder.md

```
Public Sub OpenContainingFolder()

    Dim targetWorkbook As Workbook
    Set targetWorkbook = ActiveWorkbook

    If targetWorkbook.path <> "" Then
        targetWorkbook.FollowHyperlink targetWorkbook.path
    Else
        MsgBox "Open file is not in a folder yet."
    End If

End Sub
```

PivotSetAllFields.md

```
Public Sub PivotSetAllFields()

    Dim targetTable As PivotTable
    Dim targetSheet As Worksheet

    Set targetSheet = ActiveSheet

    'this information is a bit unclear to me
    MsgBox "This defaults to the average for every Pivot table on the sheet.
        Edit code for other result."
    On Error Resume Next
    For Each targetTable In targetSheet.PivotTables
        Dim targetField As PivotField
        For Each targetField In targetTable.DataFields
            targetField.Function = xlAverage
        Next targetField
    Next targetTable

End Sub
```

SeriesSplit.md

```
Public Sub SeriesSplit()

    On Error GoTo ErrorNoSelection

    Dim selectedRange As Range
    Set selectedRange = Application.InputBox("Select category range with
       heading", Type:=8)
    Set selectedRange = Intersect(selectedRange, selectedRange.Parent.
       UsedRange).SpecialCells(xlCellTypeVisible, xlLogical + xlNumbers +
       xlTextValues)

    Dim valueRange As Range
    Set valueRange = Application.InputBox("Select values range with heading",
        Type:=8)
    Set valueRange = Intersect(valueRange, valueRange.Parent.UsedRange)

    On Error GoTo 0

    'determine default value
    Dim defaultString As Variant
    defaultString = InputBox("Enter the default value", , "#N/A")
    'strptr is undocumented
    'detect cancel and exit
    If StrPtr(defaultString) = 0 Then
        Exit Sub
    End If

    Dim dictCategories As New Dictionary

    Dim categoryRange As Range
    For Each categoryRange In selectedRange
        'skip the header row
        If categoryRange.Address <> selectedRange.Cells(1).Address Then
           dictCategories(categoryRange.Value) = 1
    Next categoryRange

    valueRange.EntireColumn.Offset(, 1).Resize(, dictCategories.Count).Insert
```

```
    'head the columns with the values

    Dim valueCollection As Variant
    Dim counter As Long
    counter = 1
    For Each valueCollection In dictCategories
        valueRange.Cells(1).Offset(, counter) = valueCollection
        counter = counter + 1
    Next valueCollection

    'put the formula in for each column
    '=IF(RC13=R1C,RC16,#N/A)
    Dim formulaHolder As Variant
    formulaHolder = "=IF(RC" & selectedRange.Column & " =R" & _
                valueRange.Cells(1).Row & "C,RC" & valueRange.Column & "," &
                    defaultString & ")"

    Dim formulaRange As Range
    Set formulaRange = valueRange.Offset(1, 1).Resize(valueRange.Rows.Count -
        1, dictCategories.Count)
    formulaRange.FormulaR1C1 = formulaHolder
    formulaRange.EntireColumn.AutoFit

    Exit Sub

ErrorNoSelection:
    'TODO: consider removing this prompt
    MsgBox "No selection made.  Exiting.", , "No selection"

End Sub
```

SeriesSplitIntoBins.md

```
Public Sub SeriesSplitIntoBins()

    Const LESS_THAN_EQUAL_TO_GENERAL As String = "<= General"
    Const GREATER_THAN_GENERAL As String = "> General"
    On Error GoTo ErrorNoSelection
```

```
6
7       Dim selectedRange As Range
8       Set selectedRange = Application.InputBox("Select category range with
            heading", Type:=8)
9       Set selectedRange = Intersect(selectedRange, selectedRange.Parent.
            UsedRange) _
10                                  .SpecialCells(xlCellTypeVisible, xlLogical +
                                        _
11                                   xlNumbers + xlTextValues)
12
13      Dim valueRange As Range
14      Set valueRange = Application.InputBox("Select values range with heading",
             Type:=8)
15      Set valueRange = Intersect(valueRange, valueRange.Parent.UsedRange)
16
17      ''need to prompt for max/min/bins
18      Dim maximumValue As Double, minimumValue As Double, binValue As Long
19
20      minimumValue = Application.InputBox("Minimum value.", "Min", _
21                                          WorksheetFunction.Min(selectedRange),
                                                Type:=1)
22
23      maximumValue = Application.InputBox("Maximum value.", "Max", _
24                                          WorksheetFunction.Max(selectedRange),
                                                Type:=1)
25
26      binValue = Application.InputBox("Number of groups.", "Bins", _
27                                        WorksheetFunction.RoundDown(Math.Sqr(
                                              WorksheetFunction.Count(selectedRange)
                                              ), _
28                                        0), Type:=1)
29
30      On Error GoTo 0
31
32      'determine default value
33      Dim defaultString As Variant
34      defaultString = Application.InputBox("Enter the default value", "Default
            ", "#N/A")
35
```

```
    'detect cancel and exit
    If StrPtr(defaultString) = 0 Then Exit Sub

    ''TODO prompt for output location

    valueRange.EntireColumn.Offset(, 1).Resize(, binValue + 2).Insert
    'head the columns with the values

    ''TODO add a For loop to go through the bins

    Dim targetBin As Long
    For targetBin = 0 To binValue
        valueRange.Cells(1).Offset(, targetBin + 1) = minimumValue + (
           maximumValue - _
                                                    minimumValue) *
                                                       targetBin / binValue
    Next

    'add the last item
    valueRange.Cells(1).Offset(, binValue + 2).FormulaR1C1 = "=RC[-1]"

    'FIRST =IF($D2 <=V$1,$U2,#N/A)
    '=IF(RC4 <=R1C,RC21,#N/A)

    'MID =IF(AND($D2 <=W$1, $D2>V$1),$U2,#N/A)  '''W current, then left
    '=IF(AND(RC4 <=R1C, RC4>R1C[-1]),RC21,#N/A)

    'LAST =IF($D2>AA$1,$U2,#N/A)
    '=IF(RC4>R1C[-1],RC21,#N/A)

    ''TODO add number format to display header correctly (helps with charts)

    'put the formula in for each column
    '=IF(RC13=R1C,RC16,#N/A)
    Dim formulaHolder As Variant
    formulaHolder = "=IF(AND(RC" & selectedRange.Column & " <=R" & _
                    valueRange.Cells(1).Row & "C," & "RC" & selectedRange.
                       Column & ">R" & _
```

```
71                      valueRange.Cells(1).Row & "C[-1]" & ")" & ",RC" &
                            valueRange.Column & "," & _
72                      defaultString & ")"
73
74      Dim firstFormula As Variant
75      firstFormula = "=IF(AND(RC" & selectedRange.Column & " <=R" & _
76                      valueRange.Cells(1).Row & "C)" & ",RC" & valueRange.
                            Column & "," & defaultString _
77                      & ")"
78
79      Dim lastFormula As Variant
80      lastFormula = "=IF(AND(RC" & selectedRange.Column & " >R" & _
81                      valueRange.Cells(1).Row & "C)" & ",RC" & valueRange.
                            Column & "," & defaultString _
82                      & ")"
83
84      Dim formulaRange As Range
85      Set formulaRange = valueRange.Offset(1, 1).Resize(valueRange.Rows.Count -
            1, binValue + 2)
86      formulaRange.FormulaR1C1 = formulaHolder
87
88      'override with first/last
89      formulaRange.Columns(1).FormulaR1C1 = firstFormula
90      formulaRange.Columns(formulaRange.Columns.Count).FormulaR1C1 =
            lastFormula
91
92      formulaRange.EntireColumn.AutoFit
93
94      'set the number formats
95
96      formulaRange.Offset(-1).Rows(1).Resize(1, binValue + 1).NumberFormat =
            LESS_THAN_EQUAL_TO_GENERAL
97      formulaRange.Offset(-1).Rows(1).Offset(, binValue + 1).NumberFormat =
            GREATER_THAN_GENERAL
98
99      Exit Sub
100
101 ErrorNoSelection:
102     'TODO: consider removing this prompt
```

```
    MsgBox "No selection made.  Exiting.", , "No selection"

End Sub
```

Sheet_DeleteHiddenRows.md

```
Public Sub Sheet_DeleteHiddenRows()
    'These rows are unrecoverable
    Dim shouldDeleteHiddenRows As VbMsgBoxResult
    shouldDeleteHiddenRows = MsgBox("This will permanently delete hidden rows
        . They cannot be recovered. Are you sure?", vbYesNo)

    If Not shouldDeleteHiddenRows = vbYes Then Exit Sub

    Application.ScreenUpdating = False

    'collect a range to delete at end, using UNION-DELETE
    Dim rangeToDelete As Range

    Dim counter As Long
    counter = 0
    With ActiveSheet
        Dim rowIndex As Long
        For rowIndex = .UsedRange.Rows.Count To 1 Step -1
            If .Rows(rowIndex).Hidden Then
                If rangeToDelete Is Nothing Then
                    Set rangeToDelete = .Rows(rowIndex)
                Else
                    Set rangeToDelete = Union(rangeToDelete, .Rows(rowIndex))
                End If
                counter = counter + 1
            End If
        Next rowIndex
    End With

    rangeToDelete.Delete

    Application.ScreenUpdating = True
```

```
32
33      MsgBox (counter & " rows were deleted")
34  End Sub
```

UnhideAllRowsAndColumns.md

```
1   Public Sub UnhideAllRowsAndColumns()
2
3       ActiveSheet.Cells.EntireRow.Hidden = False
4       ActiveSheet.Cells.EntireColumn.Hidden = False
5
6   End Sub
```