## 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
```

```
38                      matchingHeader = Application.Match(headerRow,
                            targetWorksheet.Rows(1), 0)
39
40                      'if not, add to header row
41                      If IsError(matchingHeader) Then targetWorksheet.Range("A1
                            ").End(xlToRight).Offset(, 1) = headerRow
42                  Next headerRow
43              End If
44
45              'find the PnPID column for combo
46              Dim pIDColumn As Long
47              pIDColumn = Application.Match("PnPID", targetWorksheet.Rows(1),
                    0)
48
49              'find the PnPID column for data
50              Dim pIDData As Long
51              pIDData = Application.Match("PnPID", sourceWorksheet.Rows(1), 0)
52
53              'add the data, row by row
```

```
54              Dim targetCell As Range
55              For Each targetCell In sourceWorksheet.UsedRange.SpecialCells(
                    xlCellTypeConstants)
56                  If targetCell.Row > 1 Then
57
58                      'check if the PnPID exists in the combo sheet
59                      Dim sourceRow As Variant
60                      sourceRow = Application.Match( _
61                                  sourceWorksheet.Cells(targetCell.Row, pIDData)
                                        , _
62                                  targetWorksheet.Columns(pIDColumn), _
63                                  0)
64
65                      'add new row if it did not exist and id number
66                      If IsError(sourceRow) Then
67                          sourceRow = targetWorksheet.Columns(pIDColumn).Cells(
                                targetWorksheet.Rows.Count, 1).End(xlUp).Offset(1)
                                .Row
```

```
68                          targetWorksheet.Cells(sourceRow, pIDColumn) =
                                sourceWorksheet.Cells(targetCell.Row, pIDData)
69                      End If
70
71                      'get column
72                      Dim columnNumber As Long
73                      columnNumber = Application.Match(sourceWorksheet.Cells(1,
                            targetCell.Column), targetWorksheet.Rows(1), 0)
74
75                      'update combo data
76                      targetWorksheet.Cells(sourceRow, columnNumber) =
                            targetCell
77
78                  End If
79              Next targetCell
80          End If
81      Next sourceWorksheet
82  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
```

```
9
10      Dim outputRange As Range
11      Set outputRange = Application.InputBox("Select output corner", Type:=8)
12
13      Application.ScreenUpdating = False
14      Application.EnableEvents = False
15      Application.Calculation = xlCalculationManual
16
17      Dim topLeftCell As Range
18      Set topLeftCell = sourceRange.Cells(1, 1)
19
20      Dim topRow As Long
21      topRow = topLeftCell.Row
22      Dim leftColumn As Long
23      leftColumn = topLeftCell.Column
24
25      Dim outputRow As Long
26      Dim outputColumn As Long
27      outputRow = outputRange.Row
```

```
28      outputColumn = outputRange.Column
29
30      outputRange.Activate
31
32      'Check to not overwrite
33      Dim targetCell As Range
34      For Each targetCell In sourceRange
35          If Not Intersect(sourceRange, Cells(outputRow + targetCell.Column -
            leftColumn, outputColumn + targetCell.Row - topRow)) Is Nothing
            Then
36           MsgBox ("Your destination intersects with your data. Exiting.")
37           GoTo errHandler
38          End If
39      Next
40
41      'this can be better
42      For Each targetCell In sourceRange
43          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

```

```
5       Dim selectedRange As Range
6       Set selectedRange = Application.InputBox("Select category range with
            heading", Type:=8)
7       Set selectedRange = Intersect(selectedRange, selectedRange.Parent.
            UsedRange).SpecialCells(xlCellTypeVisible, xlLogical + xlNumbers +
            xlTextValues)
8
9       Dim valueRange As Range
10      Set valueRange = Application.InputBox("Select values range with heading",
             Type:=8)
11      Set valueRange = Intersect(valueRange, valueRange.Parent.UsedRange)
12
13      On Error GoTo 0
14
15      'determine default value
16      Dim defaultString As Variant
17      defaultString = InputBox("Enter the default value", , "#N/A")
18      'strptr is undocumented
19      'detect cancel and exit
```

```
20      If StrPtr(defaultString) = 0 Then
21          Exit Sub
22      End If
23
24      Dim dictCategories As New Dictionary
25
26      Dim categoryRange As Range
27      For Each categoryRange In selectedRange
28          'skip the header row
29          If categoryRange.Address <> selectedRange.Cells(1).Address Then
                dictCategories(categoryRange.Value) = 1
30      Next categoryRange
31
32      valueRange.EntireColumn.Offset(, 1).Resize(, dictCategories.Count).Insert
33      'head the columns with the values
34
35      Dim valueCollection As Variant
36      Dim counter As Long
37      counter = 1
```

```
38      For Each valueCollection In dictCategories
39          valueRange.Cells(1).Offset(, counter) = valueCollection
40          counter = counter + 1
41      Next valueCollection
42
43      'put the formula in for each column
44      '=IF(RC13=R1C,RC16,#N/A)
45      Dim formulaHolder As Variant
46      formulaHolder = "=IF(RC" & selectedRange.Column & " =R" & _
47                  valueRange.Cells(1).Row & "C,RC" & valueRange.Column & "," &
                        defaultString & ")"
48
49      Dim formulaRange As Range
50      Set formulaRange = valueRange.Offset(1, 1).Resize(valueRange.Rows.Count -
            1, dictCategories.Count)
51      formulaRange.FormulaR1C1 = formulaHolder
52      formulaRange.EntireColumn.AutoFit
53
54      Exit Sub
```

```
55
56 ErrorNoSelection:
57     'TODO: consider removing this prompt
58     MsgBox "No selection made.  Exiting.", , "No selection"
59
60 End Sub
```

**SeriesSplitIntoBins.md**

```
1 Public Sub SeriesSplitIntoBins()
2
3     Const LESS_THAN_EQUAL_TO_GENERAL As String = "<= General"
4     Const GREATER_THAN_GENERAL As String = "> General"
5     On Error GoTo ErrorNoSelection
6
7     Dim selectedRange As Range
8     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)

    ''need to prompt for max/min/bins
    Dim maximumValue As Double, minimumValue As Double, binValue As Long

    minimumValue = Application.InputBox("Minimum value.", "Min", _
                                        WorksheetFunction.Min(selectedRange),
                                            Type:=1)

    maximumValue = Application.InputBox("Maximum value.", "Max", _
```

```
                                        WorksheetFunction.Max(selectedRange),
                                            Type:=1)

    binValue = Application.InputBox("Number of groups.", "Bins", _
                                    WorksheetFunction.RoundDown(Math.Sqr(
                                        WorksheetFunction.Count(selectedRange)
                                        ), _
                                    0), Type:=1)

    On Error GoTo 0

    'determine default value
    Dim defaultString As Variant
    defaultString = Application.InputBox("Enter the default value", "Default
        ", "#N/A")

    '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" & _
                    valueRange.Cells(1).Row & "C[-1]" & ")" & ",RC" &
                        valueRange.Column & "," & _
                    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
103     MsgBox "No selection made.  Exiting.", , "No selection"
104
```

```
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

    MsgBox (counter & " rows were deleted")
```

```
End Sub
```

### UnhideAllRowsAndColumns.md

```
Public Sub UnhideAllRowsAndColumns()


    ActiveSheet.Cells.EntireRow.Hidden = False
    ActiveSheet.Cells.EntireColumn.Hidden = False


End Sub
```