Table of Contents

[Introduction and welcome notes 2](#_Toc91483810)

[User Defined Functions 2](#_Toc91483811)

[CONCAT - concatenate string and ranges 2](#_Toc91483812)

[COUNTUNIQUE - get the count of unique values from cells, ranges, arrays, formula results. 4](#_Toc91483813)

[DAYS - Excel DAYS() function for pre 2013 Excel 5](#_Toc91483814)

[FORMULATEXT - return the absolute value of a cell 5](#_Toc91483815)

[IFS - return a value if argument is true 6](#_Toc91483816)

[IFVALUES - returns a given value if the argument is equal to a given value 7](#_Toc91483817)

[IFHYPERLINK - test cell for Hyperlink 8](#_Toc91483818)

[IFVISIBLE - a visible or hidden row mask array - include only hidden or visible rows in calculations 8](#_Toc91483819)

[MAXIFS - filter the maximum value from a range of values 9](#_Toc91483820)

[MINIFS - filter the minimum value from a range of values 11](#_Toc91483821)

[SWITCH - evaluates one value against a list of values and returns the result corresponding to the first matching value. 12](#_Toc91483822)

[SPELLNUMBER - Returns the word equivalent for a numerical number. 14](#_Toc91483823)

[SPELLNUMBERREVERSE - Returns the number equivalent for a number written as text. 17](#_Toc91483824)

[TEXTJOIN - combines the text from multiple ranges and/or strings, and includes a delimiter you specify 21](#_Toc91483825)

[TXLOOKUP - XLOOKUP for Tables/ranges using column names for dynamic column referencing 23](#_Toc91483826)

[UNIQUE - return an array of unique values, or a count of unique values 29](#_Toc91483827)

[XLOOKUP - the poor man’s version of the Microsoft XLOOKUP function for Excel 365 30](#_Toc91483828)

[Array Functions 34](#_Toc91483829)

[ARRAYIFS - IFS functionality for arrays 34](#_Toc91483830)

[ASG - array Sequence Generator - generate custom sequence arrays with ease 37](#_Toc91483831)

[CELLARRAY - return multi delimited cell(s) values as array, switch for horizontal array and/or return unique values 38](#_Toc91483832)

[CRNG - return non-contiguous ranges as contiguous for Excel functions 43](#_Toc91483833)

[FRNG - return a filtered range of values for IFS functionality in standard functions 44](#_Toc91483834)

[RETURNCOLUMNS - return chosen columns from dataset in any order, with optional limit on rows returned 46](#_Toc91483835)

[REPTX - Repeat given values to an output array. 50](#_Toc91483836)

[SEQUENCE – Microsoft’s new sequence generator 53](#_Toc91483837)

[SEQUENCER - sequence with more options, dynamic range match to other range, vertical value population in array 54](#_Toc91483838)

[SPLITIT - return element value from text array, or array location of text. 59](#_Toc91483839)

[STACKCOLUMNS - stack referenced ranges into columns of your width choice 61](#_Toc91483840)

[UNPIVOTCOLUMNS - an unpivot function. Unpivot data to an array for use in formulas or output to a table. 66](#_Toc91483841)

[VRNG - return array of columns from range as a single array 71](#_Toc91483842)

[IF Functions 72](#_Toc91483843)

[FUNCIFS - IFS criteria for all suitable functions! 72](#_Toc91483844)

[IFEQUAL - returns expected result when formula returns expected result. 74](#_Toc91483845)

[Appendix A – Links to various solutions on Reddit 76](#_Toc91483846)

[General info 76](#_Toc91483847)

[User defined functions 76](#_Toc91483848)

[365 Functions and similar 76](#_Toc91483849)

[Array functions 76](#_Toc91483850)

[IF functions 77](#_Toc91483851)

[Lookup functions 77](#_Toc91483852)

[Text return and formatting functions 77](#_Toc91483853)

[Timesheet functions 78](#_Toc91483854)

[Conditional functions 78](#_Toc91483855)

[VBA solutions 78](#_Toc91483856)

[Appendix B – Misc. Notes that apply to all UDFs 79](#_Toc91483857)

[Appendix C – non-VBA/UDF tips and tricks 79](#_Toc91483858)

[Acronyms, initialisms, abbreviations, contractions, and other phrases which expand to something larger: 79](#_Toc91483859)

[Formulas for solving interesting problems 80](#_Toc91483860)

[Formulas that deal with doing time tricks 80](#_Toc91483861)

# Introduction and welcome notes

User Defined Functions Created by /u/excelevator

<https://old.reddit.com/r/excelevator/comments/aniwgu/an_index_of_excelevator_solutions/>

Additional Functions from Better Solutions

<https://bettersolutions.com/excel/functions/index.htm>

# User Defined Functions

## CONCAT - concatenate string and ranges

CONCAT( text/range1 , [text/range2], .. )

[CONCAT](https://support.office.com/en-us/article/concat-function-9b1a9a3f-94ff-41af-9736-694cbd6b4ca2) is an Excel 365 /Excel 2019 function to concatenate text and/or range values, reproduced here for compatibility.

| **Column1** | **Column2** | **Column3** |
| --- | --- | --- |
| red | yellow | blue |
| orange |  | brown |

| **Formula** |
| --- |
| =CONCAT("Jon","Peter","Bill",A1:C2,123,456,789) |

| **Result** |
| --- |
| JonPeterBillColumn1Column2Column3redyellowblue123456789 |

For Arr[a](https://old.reddit.com/r/excel/comments/9njjz1/formula_to_just_show_numbers/e7mqug6/)ys - enter with ctrl+shift+enter

| **Return** | **FilterOut** |
| --- | --- |
| A | yes |
| B | no |
| C | no |
| D | no |

| **Formula** |
| --- |
| =CONCAT(IF(B2:B5="No",A2:A5,"")) |

| **Result** |
| --- |
| BCD |

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function CONCAT(ParamArray arguments() As Variant) As Variant

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim tmpStr As String 'build cell contents for conversion to array

Dim argType As String, uB As Double, arg As Double, cell As Variant

uB = UBound(arguments)

For arg = 0 To uB

argType = TypeName(arguments(arg))

If argType = "Range" Or argType = "Variant()" Then

For Each cell In arguments(arg)

tmpStr = tmpStr & CStr(cell)

Next

Else

tmpStr = tmpStr & CStr(arguments(arg))

End If

Next

If argType = "Error" Then

CONCAT = CVErr(xlErrNA)

Else

CONCAT = tmpStr

End If

End Function

edit 20181013 - added array functionality

edit 20191025 - minor edit for appending in line with coding recommendations

## COUNTUNIQUE - get the count of unique values from cells, ranges, arrays, formula results.

COUNTUNIQUE returns the count of unique values from all arguments.

Arguments can be values, ranges, formulas, or arrays.

Examples:

1. COUNTUNIQUE(1,1,2,3,4,"a") = 5
2. COUNTUNIQUE(A1:A6) = 5 (where the range covers the values in the first example)
3. COUNTUNIQUE(IF(A1:A10="Yes",B1:B10,"")) array formula enter with ctrl+shift+enter

There is a minor difference from the [Google sheets implementation](https://support.google.com/docs/answer/3093405) in that a null cell is rendered as 0 by the Excel parser in an array, and so is counted as the value 0. Google Sheet ignores a null value in the same scenario.

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function COUNTUNIQUE(ParamArray arguments() As Variant) As Double

'COUNTUNIQUE ( value/range/array , [value/range/array] ... ) v1.1

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

On Error Resume Next

Dim i As Double, tName As String, uB As Integer, cell As Variant

uB = UBound(arguments)

Dim coll As Collection

Dim cl As Long

Set coll = New Collection

On Error Resume Next

For i = 0 To uB

tName = TypeName(arguments(i))

If tName = "Variant()" or tName = "Range" Then

For Each cell In arguments(i)

If cell <> "" Then coll.Add cell, CStr(cell)

Next

Else

If arguments(i) <> "" Then coll.Add arguments(i), CStr(arguments(i))

End If

Next

COUNTUNIQUE = coll.Count

End Function

## DAYS - Excel DAYS() function for pre 2013 Excel

Add this function into your worksheet module.

It gives the count of days between the two dates.

Useage =DAYS([start\_date],[end\_date])

Function days(done As Long, dtwo As Long)

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Application.Volatile

Dim rtn As Long

rtn = dtwo - done

days = rtn

End Function

## FORMULATEXT - return the absolute value of a cell

FORMULATEXT ( range ) - return the absolute value in the given cell. Good for looking at formulas in cells, or the pre-formatted value.

FORMULATEXT is an Excel 2013+ function to allow easy viewing of the absolute value of a cell

See [Microsoft help](https://support.office.com/en-us/article/formulatext-function-0a786771-54fd-4ae2-96ee-09cda35439c8)

| **Cell display value** | **FORMULATEXT** | **Result** |
| --- | --- | --- |
| 19/02/2019 | =FORMULATEXT(A2) | =TODAY() |
| 20 | =FORMULATEXT(A3) | =10+10 |
| HELLO | =FORMULATEXT(A4) | =UPPER("hello") |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function FORMULATEXT(rng As Range)

FORMULATEXT = rng.Formula

End Function

## IFS - return a value if argument is true

In Excel 365/2016 Microsoft introduced the [IFS function](https://support.office.com/en-us/article/IFS-function-36329a26-37b2-467c-972b-4a39bd951d45) that is a shortener for nested IF's.

It seemed a good enough idea to develop into a UDF for lesser versions of Excel.

=IFS( arg1, arg1\_if\_true ([, arg2 , arg2\_if\_true , arg3 , arg3\_if\_true,.. ..])

See [Help file](https://support.office.com/en-us/article/IFS-function-36329a26-37b2-467c-972b-4a39bd951d45) for use.

See also similar [IFEQUAL function](https://www.reddit.com/r/excelevator/comments/5et7o1/udf_ifequal_formula_expected_result_optional_else/) for testing if values are equal.

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function IFS(ParamArray arguments() As Variant)

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim i As Long

Dim j As Long

Dim a As Long

Dim c As Integer

Dim k As Integer

i = LBound(arguments)

j = UBound(arguments)

k = (j + 1) / 2

c = 1

If WorksheetFunction.IsOdd(j + 1) Then

IFS = CVErr(xlErrValue)

End If

For a = 1 To k

If arguments(c - 1) Then

IFS = arguments(c)

Exit Function

End If

c = c + 2

Next a

IFS = CVErr(xlErrNA)

End Function

## IFVALUES - returns a given value if the argument is equal to a given value

UPDATED [here](https://old.reddit.com/r/excelevator/comments/8mwxp2/udf_switch_value_match1_return1_matchx_returnx/) with SWITCH for forward compatibility with the new Excel 365 function. Includes a default return value where no match is found and return of ranges as an option.

IFVALUES returns a given value if the argument is equal to a given value. Otherwise it returns the argument value.

Allows for test and return of multiple values entered in pairs.

Examples:

=IFVALUES( A1 , 10 ,"ten" , 20 , "twenty") 'returns "ten" if A1 is 10, "twenty" if A1 is 20, otherwise return A1

=IFVALUES( VLOOKUP( A1, B1:C20 , 2, FALSE ) , 0 , "ZERO" ) 'return "zero" when lookup is 0, other returns lookup value

Paste the following code into a [worksheet module](https://old.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function IFVALUES(arg As String, ParamArray arguments() As Variant)

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'IFVALUES ( arg , if\_value , this\_value , [if\_value, this value]..)

Dim j As Long

Dim a As Long

Dim c As Integer

Dim k As Integer

j = UBound(arguments)

k = (j + 1) / 2

c = 1

If WorksheetFunction.IsOdd(j + 1) Then

GoTo Err\_Handler

End If

For a = 1 To k

If [arg] = arguments(c - 1) Then

IFVALUES = arguments(c)

Exit Function

End If

c = c + 2

Next a

IFVALUES = [arg]

Exit Function

Err\_Handler:

IFVALUES = CVErr(xlErrValue)

End Function

## IFHYPERLINK - test cell for Hyperlink

Returns test for Hyperlink in target cell.

Use a UDF - User Defined Function.. like this one..

Copy into the worksheet Module.

1. press alt+F11
2. select your sheet from the list in the left side pane
3. From the menu, Insert Module
4. Open the Module folder for your spreadsheet and click on Module1
5. Paste the following code into the module, save.
6. Use your new function in any cell to add the same cell across all visible worksheets.
7. =isHyperlink(B15)

....

Function IsHyperlink(rng As Range)

If rng.Hyperlinks.Count = 0 Then

IsHyperlink = False

Else

IsHyperlink = True

End If

End Function

## IFVISIBLE - a visible or hidden row mask array - include only hidden or visible rows in calculations

ISVISIBLE ( range , optional hidden ) - a cell visibility array mask to exclude visible/hidden cells from formula calculations.

Where range is a single column range reference that matches the data range of your data.

Where optional hidden is 0 for a hidden values mask, and 1 is for a visible values mask. Default is 0.

This cell visibility array mask ISVISBLE UDF generates an array mask from ranges with hidden rows in the reference range that can be used in conjuction with other range arguments to include or exclude hidden or visible cells in the calculation.

For example, ISVISBLE may return an array mask of {1;0;1} where the second row is hidden, which when multiplied against a sum of array values {10,10,10} will return {10,0,10} to the equation. ([explanation here](https://old.reddit.com/r/excelevator/comments/8xblyy/arrays_and_excel_and_sumproduct/))

In the above scenario if the user opts for masking visible cells simply enter 1 as the second argument. We then have a reversed {0,1,0} mask returned.

Example: =SUMPRODUCT( ISVISBLE(A2:A10) \* (B2:B10)) returns the sum of all visible cells in B2:B10

Example2: =SUMPRODUCT( ISVISBLE(A2:A10,1) \* (B2:B10)) returns the sum of all hidden cells in B2:B10 with 1 as the second argument.

It does not really matter what theISVISBLE range column is so long as it matches the other ranges arguments in length and covers the same rows, its just using the range column reference to determine the hidden rows.

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function ISVISBLE(rng As Range, Optional hiddenCells As Boolean) As Variant

'visible mask array

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'ISVISBLE ( filtered\_range , visible/hidden)

Dim cell As Range

Dim i As Long, l As Long: l = 0

Dim booleanArray() As Boolean

On Error Resume Next

i = rng.Count - 1

ReDim booleanArray(i)

For Each cell In rng

If cell.Rows.Hidden Then

If hiddenCells Then

booleanArray(l) = True

End If

Else

If Not hiddenCells Then

booleanArray(l) = True

End If

End If

l = l + 1

Next

ISVISBLE = WorksheetFunction.Transpose(booleanArray())

End Function

## MAXIFS - filter the maximum value from a range of values

MAXIFS( max\_range , criteria\_range1 , criteria1 , [criteria\_range2, criteria2], ...)

*Title says*min\_range*, it should be*max\_range*oops! copy paste error from minifs*

[MAXIFS](https://support.office.com/en-us/article/maxifs-function-dfd611e6-da2c-488a-919b-9b6376b28883) is an Excel 365 function to filter and return the maximum value in a range, reproduced here for compatibility

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function MAXIFS(rng As Range, ParamArray arguments() As Variant) As Double

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'MAXIFS ( value\_range , criteria\_range1 , criteria1 , [critera\_range2 , criteria2]...)

Dim uB As Long, arg As Long, args As Long, cell as Range

Dim i As Long, irc As Long, l As Long, ac As Long

Dim booleanArray() As Boolean, maxifStr() As Double

On Error Resume Next

i = rng.Count - 1

ReDim booleanArray(i)

For l = 0 To i 'initialize array to TRUE

booleanArray(l) = True

Next

uB = UBound(arguments)

args = uB - 1

For arg = 0 To args Step 2 'set the boolean map for matching criteria across all criteria

l = 0

For Each cell In arguments(arg)

If booleanArray(l) = True Then

If TypeName(cell.Value2) = "Double" Then

If TypeName(arguments(arg + 1)) = "String" Then

If Not Evaluate(cell.Value2 & arguments(arg + 1)) Then

booleanArray(l) = False

End If

Else

If Not Evaluate(cell.Value = arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

Else

If Not UCase(cell.Value) Like UCase(arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

If booleanArray(l) = False Then

irc = irc + 1

End If

End If

l = l + 1

Next

Next

ReDim maxifStr(UBound(booleanArray) - irc) 'initialize array for function arguments

ac = 0

For arg = 0 To i 'use boolean map to build array for max values

If booleanArray(arg) = True Then

maxifStr(ac) = rng(arg + 1).Value 'build the value array for MAX

ac = ac + 1

End If

Next

MAXIFS = WorksheetFunction.Max(maxifStr)

End Function

## MINIFS - filter the minimum value from a range of values

MINIFS( min\_range , criteria\_range1 , criteria1 , [criteria\_range2, criteria2], ...)

[MINIFS](https://support.office.com/en-us/article/minifs-function-6ca1ddaa-079b-4e74-80cc-72eef32e6599) is an Excel 365 function to filter and return the minimum value in a range, reproduced here for compatibility.

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function MINIFS(rng As Range, ParamArray arguments() As Variant) As Double

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'MINIFS ( value\_range , criteria\_range1 , criteria1 , [critera\_range2 , criteria2]...)

Dim uB As Long, arg As Long, args As Long, cell as Range

Dim i As Long, irc As Long, l As Long, ac As Long

Dim booleanArray() As Boolean, minifStr() As Double

On Error Resume Next

i = rng.Count - 1

ReDim booleanArray(i)

For l = 0 To i 'initialize array to TRUE

booleanArray(l) = True

Next

uB = UBound(arguments)

args = uB - 1

For arg = 0 To args Step 2 'set the boolean map for matching criteria across all criteria

l = 0

For Each cell In arguments(arg)

If booleanArray(l) = True Then

If TypeName(cell.Value2) = "Double" Then

If TypeName(arguments(arg + 1)) = "String" Then

If Not Evaluate(cell.Value2 & arguments(arg + 1)) Then

booleanArray(l) = False

End If

Else

If Not Evaluate(cell.Value = arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

Else

If Not UCase(cell.Value) Like UCase(arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

If booleanArray(l) = False Then

irc = irc + 1

End If

End If

l = l + 1

Next

Next

ReDim minifStr(UBound(booleanArray) - irc) 'initialize array for function arguments

ac = 0

For arg = 0 To i 'use boolean map to build array for min values

If booleanArray(arg) = True Then

minifStr(ac) = rng(arg + 1).Value 'build the value array for MIN

ac = ac + 1

End If

Next

MINIFS = WorksheetFunction.Min(minifStr)

End Function

## SWITCH - evaluates one value against a list of values and returns the result corresponding to the first matching value.

Here is an UDF version of the [SWITCH](https://support.office.com/en-us/article/switch-function-47ab33c0-28ce-4530-8a45-d532ec4aa25e) function from Excel 2016 365.. for forward compatibility use in earlier Excel versions.

SWITCH ( Value , match\_value1 , return\_value1/range1 , [match\_value2 , return\_value2/range2 ..], [optional] defaut\_return\_value/range )

| **Formula - simple index text returns** |
| --- |
| =switch( 5, 1, "monday", 2,"tuesday", 3, "wednesday", 4,"thursday", 5,"friday", "weekend") |
| **Result** |
| Friday |
| **Formula - return different ranges based on switch values. This can be used for example, for different**VLOOKUP**ranges** |
| =VLOOKUP( "lookup\_value" , switch( "lookup\_range", "Adam",A2:B10, "Bill",C2:D10,"Jill",E2:F10,G2:H10),2,0) |
| **Result** |
| A VLOOKUP value return from the 2nd column of a table returned from SWITCH dependant on the lookup range refrence value supplied to SWITCH |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function SWITCH(arg As String, ParamArray arguments() As Variant)

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'SWITCH ( Value , match\_value1 , return\_value1 , [match\_value2 , return\_value2 ..], [optional] defaut\_return\_value )

Dim j As Long

Dim a As Long

Dim c As Integer

Dim k As Integer

j = UBound(arguments)

k = WorksheetFunction.RoundDown((j + 1) / 2, 0)

c = 1

For a = 1 To k

If [arg] = arguments(c - 1) Then

SWITCH = arguments(c)

Exit Function

End If

c = c + 2

Next a

If WorksheetFunction.IsOdd(j + 1) And IsEmpty(SWITCH) Then

SWITCH = arguments(j)

Else

SWITCH = CVErr(xlErrNA)

End If

End Function

## SPELLNUMBER - Returns the word equivalent for a numerical number.

Thanks to Bernd Plumhoff (sulprobil.com) for his contribution.  
You can use the [SPELLNUMBERREVERSE](https://bettersolutions.com/excel/functions/function-spellnumberreverse.htm) function to go in the opposite direction.

|  |
| --- |
| https://bettersolutions.com/excel/functions/function-spellnumber-example.png |

This function returns the same value for positive and negative numbers.  
All numbers will be rounded to the nearest 2 decimal places.  
This function will only return the correct text for numbers less than 999,999,999,999,999 (nine hundred trillion).  
link - http://cpap.com.br/orlando/excelspellnumbermore.asp

dbMyNumber - The number you want to convert to text.   
sMainUnitPlural - The unit to use for whole numbers.   
sMainUnitSingle - The unit to use for single whole numbers.   
sDecimalUnitPlural - (Optional) The unit to use for decimal values.   
sDecimalUnitSingle - (Optional) The unit to use for single decimal values.   
  
Public Function SPELLNUMBER(ByVal dbMyNumber As Double, \_   
                            ByVal sMainUnitPlural As String, \_   
                            ByVal sMainUnitSingle As String, \_   
                   Optional ByVal sDecimalUnitPlural As String = "", \_   
                   Optional ByVal sDecimalUnitSingle As String = "") As Variant   
  
   Dim sMyNumber As String   
   Dim sConcat As String   
   Dim sDecimalText As String   
   Dim sTemp As String   
   Dim iDecimalPlace As Integer   
   Dim iCount As Integer   
  
   ReDim Place(9) As String   
   Application.Volatile (True)   
   Place(2) = "Thousand"   
   Place(3) = "Million"   
   Place(4) = "Billion"   
   Place(5) = "Trillion"   
   sMyNumber = Trim(CStr(dbMyNumber))   
   iDecimalPlace = InStr(dbMyNumber, ".")   
  
   If iDecimalPlace > 0 Then   
      sDecimalText = GetTens(Left(Mid(Round(sMyNumber, 2), iDecimalPlace + 1) & "00", 2))   
      If Len(sDecimalText) > 0 Then   
         sMyNumber = Trim(Left(sMyNumber, iDecimalPlace - 1))   
      Else   
         sMyNumber = ""   
      End If   
   End If   
   iCount = 1   
   Do While sMyNumber <> ""   
       sTemp = GetHundreds(sMyNumber, Right(sMyNumber, 3), iDecimalPlace)   
       If sTemp <> "" Then   
          If (iCount > 1) And (LCase(Left(Trim(sConcat), 3)) <> "and") Then   
             sConcat = ", " & sConcat   
          End If   
          sConcat = sTemp & Place(iCount) & sConcat   
       End If   
       If Len(sMyNumber) > 3 Then   
           sMyNumber = Left(sMyNumber, Len(sMyNumber) - 3)   
       Else   
           sMyNumber = ""   
       End If   
       iCount = iCount + 1   
   Loop   
   Select Case Trim(sConcat)   
       Case "": sConcat = "No " & sMainUnitPlural   
       Case "One": sConcat = "One " & sMainUnitSingle   
       Case Else: sConcat = sConcat & sMainUnitPlural   
   End Select   
   If iDecimalPlace > 0 Then   
       If (Len(sDecimalUnitPlural) > 0 And Len(sDecimalUnitSingle) > 0) Then   
          sConcat = sConcat & ", "   
           Select Case Trim(sDecimalText)   
               Case "": sDecimalText = "No " & sDecimalUnitPlural   
               Case "One": sDecimalText = "One " & sDecimalUnitSingle   
               Case Else: sDecimalText = sDecimalText & sDecimalUnitPlural   
           End Select   
       Else   
       sConcat = sConcat & " and "   
       sDecimalText = Mid(Trim(Str(dbMyNumber)), iDecimalPlace + 1) & "/100"   
       End If   
   End If   
   SPELLNUMBER = Trim(sConcat & sDecimalText)   
End Function   
  
Function GetHundreds(ByVal sMyNumber As String, \_   
                     ByVal sHundredNumber As String, \_   
                     ByVal iDecimal As Integer) As String   
  
    Dim sResult As String   
      
    If sHundredNumber = "0" Then Exit Function   
    sHundredNumber = Right("000" & sHundredNumber, 3)   
    If Mid(sHundredNumber, 1, 1) <> "0" Then   
        sResult = GetDigit(Mid(sHundredNumber, 1, 1)) & "Hundred"   
    End If   
    If (sMyNumber > 1000) And (Mid(sHundredNumber, 3, 1) <> "0" Or \_   
                               Mid(sHundredNumber, 2, 1) <> "0") Or \_   
       (Len(sResult) > 0) And (Mid(sHundredNumber, 3, 1) <> "0" Or \_   
                               Mid(sHundredNumber, 2, 1) <> "0") Then   
       sResult = sResult & " and "   
    End If   
    If Mid(sHundredNumber, 2, 1) <> "0" Then   
       sResult = sResult & GetTens(Mid(sHundredNumber, 2))   
    Else   
       If Mid(sHundredNumber, 3, 1) <> "0" Then   
          sResult = sResult & GetDigit(Mid(sHundredNumber, 3))   
       Else   
          sResult = sResult & " "   
       End If   
    End If   
    GetHundreds = sResult   
End Function   
  
Function GetTens(ByVal sTensText As String) As String   
  
    Dim sResult As String   
  
    sResult = ""   
    If Left(sTensText, 1) = 1 Then   
        Select Case sTensText   
            Case "10": sResult = "Ten "   
            Case "11": sResult = "Eleven "   
            Case "12": sResult = "Twelve "   
            Case "13": sResult = "Thirteen "   
            Case "14": sResult = "Fourteen "   
            Case "15": sResult = "Fifteen "   
            Case "16": sResult = "Sixteen "   
            Case "17": sResult = "Seventeen "   
            Case "18": sResult = "Eighteen "   
            Case "19": sResult = "Nineteen "   
            Case Else   
        End Select   
    Else   
        Select Case Left(sTensText, 1)   
            Case "2": sResult = "Twenty "   
            Case "3": sResult = "Thirty "   
            Case "4": sResult = "Forty "   
            Case "5": sResult = "Fifty "   
            Case "6": sResult = "Sixty "   
            Case "7": sResult = "Seventy "   
            Case "8": sResult = "Eighty "   
            Case "9": sResult = "Ninety "   
            Case Else   
        End Select   
        sResult = sResult & GetDigit(Right(sTensText, 1))   
    End If   
    GetTens = sResult   
End Function   
  
Function GetDigit(ByVal sDigit As String) As String   
    Select Case sDigit   
        Case "1": GetDigit = "One "   
        Case "2": GetDigit = "Two "   
        Case "3": GetDigit = "Three "   
        Case "4": GetDigit = "Four "   
        Case "5": GetDigit = "Five "   
        Case "6": GetDigit = "Six "   
        Case "7": GetDigit = "Seven "   
        Case "8": GetDigit = "Eight "   
        Case "9": GetDigit = "Nine "   
        Case Else: GetDigit = ""   
    End Select   
End Function

## SPELLNUMBERREVERSE - Returns the number equivalent for a number written as text.

You can use the [SPELLNUMBER](https://bettersolutions.com/excel/functions/function-spellnumber.htm) function to go in the opposite direction.

|  |
| --- |
| https://bettersolutions.com/excel/functions/function-spellnumberreverse-example.png |

This requires a reference to the Microsoft Scripting Runtime.  
link - https://contexturesblog.com/archives/2011/10/21/words-to-numbers-in-excel/

sMyTextNumber - The text you want to convert to a number.   
  
Public Function SPELLNUMBERREVERSE( \_   
    ByVal sMyTextNumber As Variant) As Variant   
  
Dim odictionary As Scripting.Dictionary   
Dim sValidation As String   
Dim arwords As Variant   
Dim slastword As String   
Dim lmultiple As Long   
Dim lngRes As Long   
  
    On Error GoTo ErrorHandler   
    Set odictionary = StringToLong\_Dictionary   
    lmultiple = 1   
    sMyTextNumber = VBA.LCase(sMyTextNumber)   
      
    If (sMyTextNumber Like "\*,\*") Then   
        sMyTextNumber = Replace(sMyTextNumber, ",", "")   
    End If   
      
    sValidation = StringToLong\_Validation(odictionary, sMyTextNumber)   
    If (Len(sValidation) > 0) Then   
        SPELLNUMBERREVERSE = sValidation   
        Exit Function   
    End If   
      
    If (odictionary.Exists(sMyTextNumber) = True) Then   
        lngRes = odictionary.Item(sMyTextNumber)   
    Else   
        arwords = VBA.Split(sMyTextNumber, " ")   
        Do While VBA.Len(sMyTextNumber) > 0   
            slastword = arwords(UBound(arwords))   
            Select Case slastword   
                Case "and":   
                Case "hundred":   
                                 If (lmultiple = 1000) Then   
                                     lmultiple = 100000   
                                 Else: lmultiple = 100   
                                 End If   
                Case "thousand": lmultiple = 1000   
                Case Else:   
                    If (odictionary.Exists(slastword) = True) Then   
                        lngRes = lngRes + (odictionary.Item(slastword) \* lmultiple)   
                    End If   
            End Select   
            sMyTextNumber = VBA.Trim(VBA.Left(sMyTextNumber, VBA.InStrRev(sMyTextNumber, " ")))   
            arwords = VBA.Split(sMyTextNumber, " ")   
        Loop   
    End If   
  
    SPELLNUMBERREVERSE = lngRes   
    Exit Function   
      
ErrorHandler:   
    SPELLNUMBERREVERSE = "Error"   
End Function   
  
Private Function StringToLong\_Validation( \_   
    ByVal objDictionary As Scripting.Dictionary, \_   
    ByVal sMyTextNumber As Variant) As String   
      
Dim sError As String   
Dim arwords As Variant   
Dim lcount As Long   
Dim ltemp As Long   
  
    On Error GoTo ErrorHandler   
    StringToLong\_Validation = False   
          
    arwords = VBA.Split(sMyTextNumber, " ")   
    For lcount = 0 To UBound(arwords)   
        If objDictionary.Exists(arwords(lcount)) = False Then   
            sError = "Spelling mistake"   
            StringToLong\_Validation = sError   
            Exit Function   
        End If   
    Next lcount   
          
    If (VBA.InStr(1, sMyTextNumber, "thousand") > 0) Then   
        If (VBA.Right(sMyTextNumber, 8) <> "thousand") Then   
          
            If (VBA.InStr(InStr(1, sMyTextNumber, "thousand"), sMyTextNumber, "hundred") > 0) Then   
                If (VBA.InStr(1, sMyTextNumber, "thousand and") > 0) Then   
                    sError = "Invalid 'and' after the thousand"   
                    StringToLong\_Validation = sError   
                    Exit Function   
                End If   
            Else   
                If (VBA.InStr(1, sMyTextNumber, "thousand and") = 0) Then   
                    sError = "Missing 'and' after the thousand"   
                    StringToLong\_Validation = sError   
                    Exit Function   
                End If   
            End If   
        End If   
    End If   
      
    If (VBA.InStr(1, sMyTextNumber, "hundred") > 0) Then   
        If (VBA.Right(sMyTextNumber, 7) <> "hundred") Then   
            If ((VBA.InStr(1, sMyTextNumber, "hundred and") = 0) And \_   
                (VBA.InStr(1, sMyTextNumber, "hundred thousand") = 0)) Then   
                sError = "Missing 'and' after the hundred"   
                StringToLong\_Validation = sError   
                Exit Function   
            End If   
        End If   
          
        If (VBA.InStr(1, sMyTextNumber, "thousand") > 0) Then   
            sMyTextNumber = VBA.Mid(sMyTextNumber, VBA.InStr(1, sMyTextNumber, "thousand") + 9)   
        End If   
          
        If (VBA.InStr(1, sMyTextNumber, "hundred") > 0) Then   
            ltemp = VBA.InStr(1, sMyTextNumber, "hundred")   
            sMyTextNumber = VBA.Left(sMyTextNumber, ltemp + 6)   
              
            If ((sMyTextNumber <> "one hundred") And \_   
                (sMyTextNumber <> "two hundred") And \_   
                (sMyTextNumber <> "three hundred") And \_   
                (sMyTextNumber <> "four hundred") And \_   
                (sMyTextNumber <> "five hundred") And \_   
                (sMyTextNumber <> "six hundred") And \_   
                (sMyTextNumber <> "seven hundred") And \_   
                (sMyTextNumber <> "eight hundred") And \_   
                (sMyTextNumber <> "nine hundred")) Then   
                  
                sError = "You cannot have more than 9 hundreds"   
                StringToLong\_Validation = sError   
                Exit Function   
            End If   
        End If   
    End If   
    StringToLong\_Validation = ""   
    Exit Function   
      
ErrorHandler:   
    StringToLong\_Validation = sError   
End Function   
  
Private Function StringToLong\_Dictionary() As Scripting.Dictionary   
Dim objDictionary As Scripting.Dictionary   
    Set objDictionary = New Scripting.Dictionary   
    objDictionary.Add "one", 1   
    objDictionary.Add "two", 2   
    objDictionary.Add "three", 3   
    objDictionary.Add "four", 4   
    objDictionary.Add "five", 5   
    objDictionary.Add "six", 6   
    objDictionary.Add "seven", 7   
    objDictionary.Add "eight", 8   
    objDictionary.Add "nine", 9   
    objDictionary.Add "ten", 10   
    objDictionary.Add "eleven", 11   
    objDictionary.Add "twelve", 12   
    objDictionary.Add "thirteen", 13   
    objDictionary.Add "fourteen", 14   
    objDictionary.Add "fifteen", 15   
    objDictionary.Add "sixteen", 16   
    objDictionary.Add "seventeen", 17   
    objDictionary.Add "eighteen", 18   
    objDictionary.Add "nineteen", 19   
    objDictionary.Add "twenty", 20   
    objDictionary.Add "thirty", 30   
    objDictionary.Add "forty", 40   
    objDictionary.Add "fifty", 50   
    objDictionary.Add "sixty", 60   
    objDictionary.Add "seventy", 70   
    objDictionary.Add "eighty", 80   
    objDictionary.Add "ninety", 90   
      
    objDictionary.Add "hundred", -1   
    objDictionary.Add "thousand", -1   
    objDictionary.Add "and", -1   
    Set StringToLong\_Dictionary = objDictionary   
End Function

## TEXTJOIN - combines the text from multiple ranges and/or strings, and includes a delimiter you specify

Here is an UDF version of the [TEXTJOIN](https://support.office.com/en-us/article/TEXTJOIN-function-357b449a-ec91-49d0-80c3-0e8fc845691c) function from Excel 2016-365 & 2019.. for compatibility across Excel versions old and new alike.

TEXTJOIN( delimiter , ignore\_empty , "value"/range, ["value"/range]..)

=TEXTJOIN(",",TRUE,A1:D1)

| **Column1** | **Column2** | **Column3** |
| --- | --- | --- |
| red | yellow | blue |
| orange |  | brown |

| **Formula** |
| --- |
| =TEXTJOIN(",",TRUE,"Jon","Peter","Bill",A1:C2,123,456,789) |
| **Result** |
| Jon,Peter,Bill,Column1,Column2,Column3,red,yellow,blue,orange,brown,123,456,789 |

For Arr[a](https://old.reddit.com/r/excel/comments/8m426v/concatenate_all_cells_with_matching_number_in/dzm8hlp/)ys - enter with ctrl+shift+enter

| **Return** | **FilterOut** |
| --- | --- |
| A | yes |
| B | no |
| C | no |
| D | no |

| **Formula** |
| --- |
| =TEXTJOIN(",",TRUE,IF(B2:B5="No",A2:A5,"")) |
| **Result** |
| B,C,D |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function TEXTJOIN(delim As String, ie As Boolean, ParamArray arguments() As Variant) As Variant 'v2\_02

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'TEXTJOIN( delimiter , ignore\_empty , "value"/range, ["value"/range]..)

'See Microsoft TEXTJOIN Helpfile

Dim tmpStr As String 'build cell contents for conversion to array

Dim argType As String, uB As Double, arg As Double, cell As Variant

uB = UBound(arguments)

For arg = 0 To uB

argType = TypeName(arguments(arg))

If argType = "Range" Or argType = "Variant()" Then

For Each cell In arguments(arg)

If ie = True And cell = "" Then

'do nothing

Else

tmpStr = tmpStr & CStr(cell) & delim

End If

Next

Else

If ie = True And CStr(arguments(arg)) = "" Then

'do nothing

Else

tmpStr = tmpStr & CStr(arguments(arg)) & delim

End If

End If

Next

If argType = "Error" Then

TEXTJOIN = CVErr(xlErrNA)

Else

tmpStr = IIf(tmpStr = "", delim, tmpStr) 'fill for no values to avoid error below

TEXTJOIN = Left(tmpStr, Len(tmpStr) - Len(delim))

End If

End Function

edit: 16/05/2018 Added array functionality - let me know if you find a bug!

edit: 28/05/2018 Added ignore blank for string input

edit: 10/06/2018 Complete re-write after overnight epiphany

edit: 11/12/2018 Fixed where an error was returned on blank value set of cells, now returns blank

edit: 29/09/2019 Fixed error with no return v2.01

edit: 25/10/2019 - minor edit for appending in line with coding recommendations

edit: known  issue, returns 0 for an empty cell value in array IF function. The array returns 0, not my code... Blank cells in Excel are consider to contain a FALSE value which is rendered as 0 behind the scenes.

## TXLOOKUP - XLOOKUP for Tables/ranges using column names for dynamic column referencing

TXLOOKUP ( value , table/range, search\_col, return\_values , [match\_type] , [search\_type])

06/02/2020: Please note A re-write of this UDF is in progress due to issues in the current structure in dealing with the different range and text references causing an 1 line offset in certain circumstances.

No more INDEX(MATCH,MATCH) or XLOOKUP(XLOOKUP) or VLOOKUP(MATCH/CHOOSE) or any other combination to dynamically lookup columns from tables.

TXLOOKUP takes table and column arguments to dynamically search and return those columns you reference by name.

TXLOOKUP can return single values or contiguous result cells from the result column as an array formula

TXLOOKUP was built to resemble the new XLOOKUP function from Microsoft for similarity.

The TXLOOKUP parameters are as follows:

1. Value - the lookup value, either as a Text value and/or a cell reference and/or combination of functions.
2. Table - the Table or cell range reference to the table of data to use for the lookup
3. Lookup\_col - the name of the column to lookup the value in, either as a Text value or a cell reference or combination of functions.
4. Return\_cols - the column or range of columns to return data from where a match has been found for the lookup value on that row.
5. Match\_type (optional) as per XLOOKUP
6. Search\_type (optional) as per XLOOKUP

TXLOOKUP has been written to ease the lookup of Tables where finding the column index, or understanding the additional formulas for lookup values. Here are some features:

1. Can use Table references, Text, or range references in the arguments
2. The naming of columns makes for a dynamic formula unreliant on column position
3. Shares the parameters of XLOOKUP so as to compliment XLOOKUP
4. Can return the whole row or a contigous ranges of cells of the return row.

Lookup type arguments are the same as XLOOKUP

**match\_type**

0 exact match - done by default

-1 exact match or next smaller item

1 exact match or next larger item

2 wildcard character match

**search\_type**

-1 search last to first

1 search first to last

2 binary search sorted ascending order

-2 binsary search sorted descending order

Examples

The types of addressing are interchangeable in the formula, using Table, or cell, or Text/Number value referencing.

Example formula for a product table PTable

1. =TXLOOKUP ( A1 , PTable , "ItemID" , "ItemDesc")
2. =TXLOOKUP ( A1 & "123" , PTable , PTable[[#Headers],[ItemID]] , PTable)
3. =TXLOOKUP ( A1 & "123" , PTable , "ItemID" , PTable[[ItemDesc]:[ItemPrice]])
4. =TXLOOKUP ( "ABC123" , A1:E250 , "ItemID" , A1:E1)
5. =TXLOOKUP ( "ABC123" , A1:E250 , "ItemID" , "ItemDesc:ItemPrice")

Source table for examples, named Table1 at A1:E6

| **ID** | **Name** | **Address** | **Age** | **Sex** |
| --- | --- | --- | --- | --- |
| 101 | Andrew Smith | 1 Type St, North State | 55 | M |
| 102 | Robert Anderson | 15 Jerricho Place, South State | 16 | M |
| 103 | Peter Duncan | 77 Ark Pl, Western Place | 27 | M |
| 104 | Julia Fendon | 22 Ichen Street, North State | 33 | F |
| 105 | Angela Keneally | 66 Pelican Avenue, East Place | 43 | F |

Examples

Lookup Client ID and return the client name column from table

Reference in Table format or plain text or cell reference of column name

=TXLOOKUP ( 103 , Table1 , Table1[[#Headers],[ID]] , Table1[Name])

Or =TXLOOKUP ( 103 , Table1 , "ID" , "Name")

Or =TXLOOKUP ( A4 , A1:E6 , "ID" , "Name")

Result Peter Duncan

Return the table row that holds the search value. Requires array formula across cells to return all values. Enter with ctrl+shift+enter.

=TXLOOKUP ( 103 , Table1 , "ID" , Table1)

Result 103 | Peter Duncan | 77 Ark Pl, Western Place | 27 | M

Return Name, Address, and Age from row. Requires array formula across cells to return all values. Enter with ctrl+shift+enter.

=TXLOOKUP ( 103 , Table1 , Table1[[#Headers],[ID]] , Table1[[Name]:[Age])

Or =TXLOOKUP ( A4 , Table1 , "ID" , "Name:Age")

Or =TXLOOKUP ( 103 , A1:E6 , "ID" , "Name:Age")

Result Peter Duncan | 77 Ark Pl, Western Place | 27

Return the name of the last male identity in the table, searching last to first

=TXLOOKUP ( "M" , Table1 , "Sex", "Name" , 0 , -1)

Result Peter Duncan

Return the Name and Address of the person living in Ichen street. Requires array formula across cells to return all values. Enter with ctrl+shift+enter.

=TXLOOKUP ( "\*Ichen\*" , Table1 , "Address", Table1[[Name]:[Address]] , 2 )

Result Julia Fendon | 22 Ichen Street, North State

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function TXLOOKUP(sVal As Variant, tblRng As Variant, cRng As Variant, rtnVals As Variant, Optional arg1 As Variant, Optional arg2 As Variant) As Range 'v1.06

'TXLOOKUP ( value , table/range, search\_col, return\_values , [match\_type] , [search\_type])

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

If IsMissing(arg1) Then arg1 = 0

If IsMissing(arg2) Then arg2 = 0

Dim rsult As Variant 'take the final result array

Dim srchRng As Range 'the search column range

Dim rtnRng As Range 'the return column range

Dim srchVal As Variant: srchVal = sVal '.Value 'THE SEARCH VALUE

Dim sIndex As Double: sIndex = tblRng.Row - 1 'the absolute return range address

Dim n As Long 'for array loop

'format the search value for wildcards or not

If (arg1 <> 2 And VarType(sVal) = vbString) Then srchVal = Replace(Replace(Replace(srchVal, "\*", "~\*"), "?", "~?"), "#", "~#") 'for wildcard switch, escape if not

'-----------------------

Dim srchType As String

Dim matchArg As Integer

Dim lDirection As String

Dim nextSize As String

Select Case arg1 'work out the return mechanism from parameters, index match or array loop

Case 0, 2

If arg2 = 0 Or arg2 = 1 Then

srchType = "im"

matchArg = 0

End If

Case 1, -1

nextSize = IIf(arg1 = -1, "s", "l") 'next smaller or larger

If arg2 = 0 Or arg2 = 1 Then

srchType = "lp"

lDirection = "forward"

End If

End Select

Select Case arg2 'get second parameter processing option

Case -1

srchType = "lp": lDirection = "reverse"

Case 2

srchType = "im": matchArg = 1

Case -2

srchType = "im": matchArg = -1

End Select

'sort out search and return ranges

Dim hdrRng As Range 'search range for header return column

If tblRng.ListObject Is Nothing Then 'is it a table or a range

Set hdrRng = tblRng.Rows(1)

Set srchRng = tblRng.Columns(WorksheetFunction.Match(cRng, hdrRng, 0)) 'set the search column range

Else

Set hdrRng = tblRng.ListObject.HeaderRowRange

Set srchRng = tblRng.ListObject.ListColumns(WorksheetFunction.Match(cRng, hdrRng, 0)).Range

End If

Set srchRng = srchRng.Resize(srchRng.Rows.Count - 1).Offset(1, 0) 'remove header from range

'get column to search

Dim rtnValsType As String: rtnValsType = TypeName(rtnVals)

Select Case rtnValsType

Case "String"

If InStr(1, rtnVals, ":") Then

Dim args() As String, iSt As Double, iCd As Double, rsz As Double

args = Split(rtnVals, ":")

iSt = WorksheetFunction.Match(args(0), hdrRng, 0)

iCd = WorksheetFunction.Match(args(1), hdrRng, 0)

rsz = iCd - iSt + 1

Set rtnRng = tblRng.Columns(WorksheetFunction.Match(args(0), hdrRng, 0)).Resize(srchRng.Rows.Count, rsz)

Else

Set rtnRng = tblRng.Columns(WorksheetFunction.Match(rtnVals, hdrRng, 0)).Resize(srchRng.Rows.Count).Offset(1, 0)

End If

Case "Range"

If rtnVals.ListObject Is Nothing And rtnVals.Count = 1 Then 'set the return range

Set rtnRng = tblRng.Columns(WorksheetFunction.Match(rtnVals, hdrRng, 0))

If tblRng.ListObject Is Nothing Then Set rtnRng = rtnRng.Resize(srchRng.Rows.Count).Offset(1, 0)

ElseIf rtnVals.Rows.Count <> tblRng.Rows.Count Then 'assume header name only reference

Set rtnRng = rtnVals.Resize(srchRng.Rows.Count, rtnVals.Columns.Count)

Set rtnRng = rtnRng.Resize(srchRng.Rows.Count).Offset(1, 0)

Else

If Not rtnVals.ListObject Is Nothing Then

Set rtnRng = rtnVals.Resize(srchRng.Rows.Count, rtnVals.Columns.Count)

Else

Set rtnRng = rtnVals ' return the table

Set rtnRng = rtnRng.Resize(srchRng.Rows.Count).Offset(1, 0)

End If

End If

End Select

'start the searches

If srchType = "im" Then ' for index match return

Set TXLOOKUP = rtnRng.Rows(WorksheetFunction.Match(srchVal, srchRng, matchArg))

Exit Function

Else 'load search range into array for loop search

Dim vArr As Variant: vArr = srchRng 'assign the lookup range to an array

Dim nsml As Variant: ' nsmal - next smallest value

Dim nlrg As Variant: ' nlrg - next largest value

Dim nStart As Double: nStart = IIf(lDirection = "forward", 1, UBound(vArr))

Dim nEnd As Double: nEnd = IIf(lDirection = "forward", UBound(vArr), 1)

Dim nStep As Integer: nStep = IIf(lDirection = "forward", 1, -1)

For n = nStart To nEnd Step nStep

If vArr(n, 1) Like srchVal Then Set TXLOOKUP = rtnRng.Rows(n): Exit Function 'exact match found

If nsml < vArr(n, 1) And vArr(n, 1) < srchVal Then 'get next smallest

Set nsml = srchRng.Rows(n)

End If

If vArr(n, 1) > srchVal And (IsEmpty(nlrg) Or nlrg > vArr(n, 1)) Then 'get next largest

Set nlrg = srchRng.Rows(n)

End If

Next

End If

If arg1 = -1 Then 'next smallest

Set TXLOOKUP = rtnRng.Rows(nsml.Row - sIndex)

ElseIf arg1 = 1 Then 'next largest

Set TXLOOKUP = rtnRng.Rows(nlrg.Row - sIndex)

End If

End Function

## UNIQUE - return an array of unique values, or a count of unique values

[UNIQUE](https://support.office.com/en-us/article/unique-function-c5ab87fd-30a3-4ce9-9d1a-40204fb85e1e) has arrived for Excel 365.

Reproduced here for all - though the optional count switch here is not in the Microsoft version.

UNIQUE will return an array of unique values or a count of unique values.

Use =UNIQUE ( range , [optional] 0/1 )

0 returns an array of unique values, 1 returns a count of unique values. 0 is the default return.

Example use returning a unique list of value to TEXTJOIN for delimited display

=TEXJOIN(",",TRUE,UNIQUE(A1:A50)

Example use returning a count of unique values

=UNIQUE(A1:A50 , 1 )

Example returning a unique list filtered against other field criteria; entered as array formula ctrl+shift+enter

=TEXTJOIN(",",TRUE,UNIQUE(IF(A1:A50="Y",B1:B50,"")))

Example returning the count of unique values from a list of values. UNIQUE expects a comma delimited list of values in this example to count the unique values.

=UNIQUE(TEXTIFS(C1:C12,",",TRUE,A1:A12,"A",B1:B12,"B"),1)

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function UNIQUE(RNG As Variant, Optional cnt As Boolean) As Variant

'UNIQUE ( Range , [optional] 0 array or 1 count of unique ) v1.2.3

'http://reddit.com/u/excelevator

'http://reddit.com/r/excelevator

If IsEmpty(cnt) Then cnt = 0 '0 return array, 1 return count of unique values

Dim i As Long, ii As Long, colCnt As Long, cell As Range

Dim tName As String: tName = TypeName(RNG)

If tName = "Variant()" Then

i = UBound(RNG)

ElseIf tName = "String" Then

RNG = Split(RNG, ",")

i = UBound(RNG)

tName = TypeName(RNG) 'it will change to "String()"

End If

Dim coll As Collection

Dim cl As Long

Set coll = New Collection

On Error Resume Next

If tName = "Range" Then

For Each cell In RNG

coll.Add Trim(cell), Trim(cell)

Next

ElseIf tName = "Variant()" Or tName = "String()" Then

For ii = IIf(tName = "String()", 0, 1) To i

coll.Add Trim(RNG(ii)), Trim(RNG(ii))

coll.Add Trim(RNG(ii, 1)), Trim(RNG(ii, 1))

Next

End If

colCnt = coll.Count

If cnt Then

UNIQUE = colCnt

Else

Dim lp As Long

Dim rtnArray() As Variant

ReDim rtnArray(colCnt - 1)

For lp = 1 To colCnt

rtnArray(lp - 1) = coll.Item(lp)

Next

UNIQUE = WorksheetFunction.Transpose(rtnArray)

End If

End Function

Let me know if you find any bugs

edit 08/04/2019 - v1.2 - accept text list input from other functions, expects comma delimited values

edit 12/04/2019 - v1.2.1 - corrected i count for array

edit 21/04/2019 - v1.2.2 - corrected i count for array again. Was erroring on typneame count with wrong start index

edit 16/09/2021 v1.2.3 - return vertical array in line with Excel 365 function. Did not realise it was returning a horizontal array

## XLOOKUP - the poor man’s version of the Microsoft XLOOKUP function for Excel 365

UPDATED with **IF\_NOT\_FOUND** argument which was added after the initial review release of **XLOOKUP**

XLOOKUP ( value , lookup\_range , return\_range , [if\_not\_found], [match\_type] , [search\_type])

This UDF was built for people to experience the [new XLOOKUP function](https://techcommunity.microsoft.com/t5/Excel-Blog/Announcing-XLOOKUP/ba-p/811376) from Microsoft, in versions of Excel that do not have access to that function.

Being a UDF written in VBA for older Excel versions it will not be as quick or efficient as the native version. For that I encourage you to upgrade your software.

This UDF offers the chance to have a play with the new functionality, and offers compatibility for versions (without accepting arrays as the range arguments and as value search arguments), still working on that which is multi-range and multi-cell value array functionality.

The functionality in this UDF is taken from what I have seen to date on the XLOOKUP functions press releases and from the links below covering the new function;

[Microsoft - XLOOKUP function](https://support.office.com/en-us/article/xlookup-function-b7fd680e-6d10-43e6-84f9-88eae8bf5929)

[Microsoft Techcommunity XLOOKUP announcement with examples](https://techcommunity.microsoft.com/t5/Excel-Blog/Announcing-XLOOKUP/ba-p/811376)

[Bill Jelen MVP - The VLOOKUP Slayer: XLOOKUP Debuts Excel](https://www.mrexcel.com/excel-tips/the-vlookup-slayer-xlookup-debuts-excel/)

[Bill Jelen MVP - XLOOKUP in Excel is VLOOKUP Slayer Video](https://www.youtube.com/watch?v=E5JxX_3Qb7A)

[BIll Jelen MVP - XLOOKUP or INDEX-MATCH-MATCH Head-to-Head Video](https://www.youtube.com/watch?v=8dwmvTka3gs)

Important note

To view the array functionality, select the range of cells to hold the array and enter the formula with ctrl+shift+enter to see it populate across the cells. Those of you with the dynamic array version of Excel should see the expansion without ctrl+shift+enter.

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function XLOOKUP(searchVal As Variant, searchArray As Range, returnArray As Variant, Optional notFound As Variant, Optional arg1 As Variant, Optional arg2 As Variant) As Variant 'v1.1

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

If IsMissing(arg1) Then arg1 = 0

If IsMissing(arg2) Then arg2 = 0

Dim rsult As Variant 'take the final result array

Dim r2width As Integer: r2width = searchArray.Columns.Count

Dim r3width As Integer: r3width = returnArray.Columns.Count

Dim rtnHeaderColumn As Boolean: rtnHeaderColumn = r2width > 1

If r2width > 1 And r2width <> r3width Then

XLOOKUP = CVErr(xlErrRef)

Exit Function

End If

Dim srchVal As Variant: srchVal = searchVal 'THE SEARCH VALUE

Dim sIndex As Double: sIndex = searchArray.Row - 1 'the absolute return range address

Dim n As Long 'for array loop

'format the search value for wildcards or not

If (arg1 <> 2 And VarType(searchVal) = vbString) Then srchVal = Replace(Replace(Replace(srchVal, "\*", "~\*"), "?", "~?"), "#", "~#") 'for wildcard switch, escape if not

'-----------------------

Dim srchType As String

Dim matchArg As Integer

Dim lDirection As String

Dim nextSize As String

On Error GoTo error\_control

Select Case arg1 'work out the return mechanism from parameters, index match or array loop

Case 0, 2

If arg2 = 0 Or arg2 = 1 Then

srchType = "im"

matchArg = 0

End If

Case 1, -1

nextSize = IIf(arg1 = -1, "s", "l") 'next smaller or larger

If arg2 = 0 Or arg2 = 1 Then

srchType = "lp"

lDirection = "forward"

End If

End Select

Select Case arg2 'get second parameter processing option

Case -1

srchType = "lp": lDirection = "reverse"

Case 2

srchType = "im": matchArg = 1

Case -2

srchType = "im": matchArg = -1

End Select

If srchType = "im" Then ' for index match return

If rtnHeaderColumn Then

Set XLOOKUP = returnArray.Columns(WorksheetFunction.Match(srchVal, searchArray, matchArg))

Else

Set XLOOKUP = returnArray.Rows(WorksheetFunction.Match(srchVal, searchArray, matchArg))

End If

Exit Function

Else 'load search range into array for loop search

Dim vArr As Variant: vArr = IIf(rtnHeaderColumn, WorksheetFunction.Transpose(searchArray), searchArray) 'assign the lookup range to an array

Dim nsml As Variant: ' nsmal - next smallest value

Dim nlrg As Variant: ' nlrg - next largest value

Dim nStart As Double: nStart = IIf(lDirection = "forward", 1, UBound(vArr))

Dim nEnd As Double: nEnd = IIf(lDirection = "forward", UBound(vArr), 1)

Dim nStep As Integer: nStep = IIf(lDirection = "forward", 1, -1)

For n = nStart To nEnd Step nStep

If vArr(n, 1) Like srchVal Then Set XLOOKUP = IIf(rtnHeaderColumn, returnArray.Columns(n), returnArray.Rows(n)): Exit Function 'exact match found

If nsml < vArr(n, 1) And vArr(n, 1) < srchVal Then 'get next smallest

Set nsml = searchArray.Rows(n)

End If

If vArr(n, 1) > srchVal And (IsEmpty(nlrg) Or nlrg > vArr(n, 1)) Then 'get next largest

Set nlrg = IIf(rtnHeaderColumn, searchArray.Columns(n), searchArray.Rows(n))

End If

Next

End If

If arg1 = -1 Then 'next smallest

Set XLOOKUP = returnArray.Rows(nsml.Row - sIndex)

ElseIf arg1 = 1 Then 'next largest

Set XLOOKUP = returnArray.Rows(nlrg.Row - sIndex)

End If

If Not IsEmpty(XLOOKUP) Then Exit Function

error\_control:

If IsMissing(notFound) Then

XLOOKUP = CVErr(xlErrNA)

Else

XLOOKUP = [notFound]

End If

End Function

Let me know of any bugs

20190915: v1. I now see that the [official XLOOKUP](https://support.office.com/en-us/article/xlookup-function-b7fd680e-6d10-43e6-84f9-88eae8bf5929) version does array formulas with concatenation of cells and ranges; at this stage the UDF above does not do that.. I am thinking about how to get that happening as it introduces a bit of a coding challenge.

20190916: v1.01. removed errant r3width value assignment

20190917: v1.02. srchVal from = rng1.Value to rng1 as was causing error with number entry

20190918 - there are a couple of issues that I am working on, accepting arrays as the range arguments and as value search arguments. These are issues that are not really part of the everyday use of the function, and are for more advanced uses.

20201207- Added the IF\_NOT\_FOUND argument

# Array Functions

## ARRAYIFS - IFS functionality for arrays

ARRAYIFS is an experiment in adding IFS functionality for arrays passed into the function.

ARRAYIFS ( function , data\_column , array , col1 , arg1 [, col2 , arg2 ] .. )

ARRAYIFS ( "stdev" , 3 , data\_array , 1 , ">0" , 2 , "johns\_data" )

ARRAYIFS was developed after the creation of [STACKCOLUMNS](https://old.reddit.com/r/excelevator/comments/b01vxx/udf_stackcolumns_column_stack_width_range1_range2/), [RETURNCOLUMNS](https://old.reddit.com/r/excelevator/comments/awhi9v/udf_returncolumns_row_limit_range_col1_col2/), and [UNPIVOTCOLUMNS](https://old.reddit.com/r/excelevator/comments/b0n4v0/udf_unpivotcolumns_range_column_name_col1range1/) after realising it would not be easy to use those array functions in standard Excel functions as the data source.

I had no idea of the kind of processing speed to expect, suffice to say it is very slow comparitive to native range functions.

The arguments:

function is the function to apply to the data. The list of functions available can be seen at the bottom of the code. More functions can be added by the user as required, though they are limited to single dimension arrays.

data\_column is the index of the column in the passed array to apply the function to.

array is the array of data to pass to the function.

col1 is the column to apply the filter argument to.

arg1 is the argument to apply to the assosiated column

Note the Excel VBA array limit of 65536 rows of data applies to this UDF in older versions - just be aware

Example

Join 2 tables with [STACKCOLUMNS](https://old.reddit.com/r/excelevator/comments/b01vxx/udf_stackcolumns_column_stack_width_range1_range2/) and sum values in column 2 where column 1 values = "UK"

=ARRAYIFS("sum",2,stackcolumns(2,Table1,Table2),1,"UK")

| **Country** | **Value** |
| --- | --- |
| UK | 10 |
| US | 20 |
| UK | 30 |
| US | 40 |
|  |  |
| **Country** | **Value** |
| UK | 1 |
| US | 2 |
| UK | 3 |
| US | 4 |
|  |  |
| **Answer** | **44** |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function ARRAYIFS(func As String, wCol As Integer, rng As Variant, ParamArray arguments() As Variant) As Double

'ARRAYIFS ( function , column , array , col1 , arg1 [ ,col2, arg2].. )

'ARRAYIFS ( "sum" , 3 , unpivotdata() , 1 , "January" , 2 , ">0" ) )

Dim uB As Double, arg As Double, args As Double, arrayLen As Double, i As Double, l As Double, j As Double, ac As Double, irc As Double 'include row count to initialize arrya

Dim booleanArray() As Variant

Dim valueArray() As Double

arrayLen = UBound(rng) - 1

ReDim booleanArray(arrayLen)

For l = 0 To arrayLen 'initialize array to TRUE

booleanArray(l) = True

Next

uB = UBound(arguments)

args = uB - 1

For arg = 0 To args Step 2 'set the boolean map for matching criteria across all criteria

For j = 0 To arrayLen 'loop through each array element of the passed array

If booleanArray(j) = True Then

If TypeName(rng(j + 1, arguments(arg))) = "Double" Then

If TypeName(arguments(arg + 1)) = "String" Then

If Not Evaluate(rng(j + 1, arguments(arg)) & arguments(arg + 1)) Then

booleanArray(j) = False

End If

Else

If Not Evaluate(rng(j + 1, arguments(arg)) = arguments(arg + 1)) Then

booleanArray(j) = False

End If

End If

Else

If Not UCase(rng(j + 1, arguments(arg))) Like UCase(arguments(arg + 1)) Then

booleanArray(j) = False

End If

End If

If booleanArray(j) = False Then

irc = irc + 1

End If

End If

Next

Next

ReDim valueArray(UBound(booleanArray) - irc) 'initialize array for function arguments

ac = 0

For arg = 0 To arrayLen 'use boolean map to build array

If booleanArray(arg) = True Then

valueArray(ac) = rng(arg + 1, wCol)

ac = ac + 1

End If

Next

Select Case LCase(func) 'add functions as required here

Case "sum": ARRAYIFS = WorksheetFunction.Sum(valueArray)

Case "stdev": ARRAYIFS = WorksheetFunction.StDev(valueArray)

Case "average": ARRAYIFS = WorksheetFunction.Average(valueArray)

Case "count": ARRAYIFS = WorksheetFunction.Count(valueArray)

'Case "NAME HERE": ARRAYIFS = WorksheetFunction.NAME\_HERE(valueArray) '<==Copy, Edit, Uncomment

End Select

End Function

## ASG - array Sequence Generator - generate custom sequence arrays with ease

UDF - ASG ( startNum , endNum , step )

One of the difficulties in generating complex array results is getting the array seeding sequence into a usable format.

ASG - Array Sequence Generator allows for easy generation of custom complex steps of values.

Each parameter can take a value or formula. The default step value is 1.

Example1: We want all values between 1 and 5 at intervals of 1

=ASG(1,5) returns { 1 , 2 , 3 , 4 , 5}

Example2: We want all values between -5 and -25 at intervals of -5

=ASG(-5,-25,-5) returns { -5 , -10 , -15 , -20 , -25 }

Example3: We want all values for the row count of a 10 row range Table1[Col1] at intervals of 2

=ASG(1,COUNTA(Table1[Col1]),2) returns { 1, 3 , 5 , 7 , 9 }

Example4: We want all value between -16 and 4 at intervals of 4.5

=ASG(-16,4,4.5) returns { -16 , -11.5 , -7 , -2.5 , 2 }

Example5: We want all values between 0 and Pi at intervals of .557

=ASG(0.1,Pi(),0.557) returns {0.1, 0.657 , 1.214 , 1.771 , 2.328 , 2.885 }

If you need the array in horizonal format then wrap ASG in TRANSPOSE

=TRANSPOSE(ASG(1,5)) returns { 1 ; 2 ; 3 ; 4 ; 5}

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function ASG(sNum As Double, enNum As Double, Optional nStep As Double) As Variant

'ASG - Array Sequence Genetator; generate any desired array sequence

'ASG ( StartNumber , EndNumber , optional ValueStep )

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

If nStep = 0 Then

nStep = 1 'default step is 1

End If

Dim rArray() As Double

Dim i As Double, j As Double: j = 0

ReDim rArray(WorksheetFunction.RoundDown(Abs(sNum - enNum) / Abs(nStep), 0))

For i = sNum To enNum Step nStep

rArray(j) = Round(i, 10)

j = j + 1

i = Round(i, 10) ' to clear up Excel rounding error and interuption of last loop on occasion

Next

ASG = rArray()

End Function

## CELLARRAY - return multi delimited cell(s) values as array, switch for horizontal array and/or return unique values

CELLARRAY will return an array of values from the reference cell(s) or text array. The array being anything the user determines is splitting the text into elements of an array.

CELLARRAY can return a unique set of values from input data using the /u switch.

CELLARRAY can return a horizontal or vertical array.

Use: =CELLARRAY( range, \*delimiter[s], [optional] "/h", [optional] "/u")

range is the reference range or text value. A multi cell range can be selected for addition to the array output.

delimiter[s] is whatever you determine that delimits the text array elements. Multiple delimiters can be expressed. Spaces are trimmed from the source data. \*This value is not required where the range is just a range of cells.

"/h" will deliver a horizontal array. Vertical is the default.

"/u" will return a unique set of values where duplicates exist in the input values.

Examples (ctrl+shift+enter)

=CELLARRAY ( A1 , "/", ":","," ) returns {1,2,3,4} where A1 = 1,2/3:4

=CELLARRAY ( A1 , "/", ":","," ,"/h") returns {1;2;3;4} where A1 = 1,2/3:4

=CELLARRAY ( A1 , "/", ":","," , "/u" ) returns {1,2,3,4} where A1 = 1,1,2/3:4:4

=CELLARRAY ( "192.168.11.12" , "." ) returns {192,168,11,12}

=CELLARRAY ( "5 - 6 - 7 - 8" , "-" ) returns {5,6,7,8}

=CELLARRAY ( "A1:A5" ) returns {1,2,3,4,5} where A1:A5 is 1 to 5 respectively

=CELLARRAY("Sun/Mon/Tue/Wed/Thu/Fri/Sat","/")) returns {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}

Examples in functions (ctrl+shift+enter)

=SUM(cellarray("36, 52, 29",",")\*1) returns 117

=SUM(cellarray(A1,":")\*1) returns 117 where A1 = 36 :52: 29

Multi cell with multi delimiter processing - select the cells, paste at A1

| **Formula** | **values** |
| --- | --- |
| ="Answer: "&SUM( cellarray(B2:B4,",",":",";","/")) | 1 ,2 ; 3 / 4 : 5 |
| Answer: 105 | 6,7,8;9 |
|  | 10, 11 , 12 /13;14 |

Use the /h horizontal switch to transpose the array - select the cells, enter the formula in the first cell and ctrl+shift+enter

| **Formula** | **value** |  |
| --- | --- | --- |
| =cellarray(B2,",","/h") | 36, 52, 29 |  |
| 36 | 52 | 29 |

Default vertical return - select the cells, enter the formula in the first cell and ctrl+shift+enter

| **Formula** | **value** |
| --- | --- |
| =cellarray(B1,","") | 36, 52, 29 |
| 36 |  |
| 52 |  |
| 29 |  |

Text array - select the cells, use the /u unique switch to return unique values, enter the formula in the first cell and ctrl+shift+enter

| **Formula** | **values** |
| --- | --- |
| =cellarray(B2,",", "/u") | hello, hello, how, how , are, are, you, you |
| hello |  |
| how |  |
| are |  |
| you |  |

CELLARRAY can also be used in conjunction with [TEXTIFS](https://www.reddit.com/r/excelevator/comments/5rfriw/udf_textifs_return_range_delimiter_ignore_blanks/) to generate dynamic cell range content of **unique filtered** values .

Example use;

| **Type** | **Item** | **Fruit** |
| --- | --- | --- |
| Fruit | apple | =IFERROR(CELLARRAY(TEXTIFS(B2:B8,",",TRUE,A2:A8,C1),",","/u"),"") |
| Fruit | banana | banana |
| Fruit | berry | berry |
| Fruit | berry | lime |
| Metal | iron |  |
| Fruit | lime |  |
| Metal | silver |  |

Copy the table above to A1:B8

Highlight **C2:C8 and copy the following formula into the formula bar** and press **ctrl+shfit+enter** , the formula is entered as a cell array. The /u switch ensure the return of unique values only

=IFERROR(CELLARRAY(TEXTIFS(B2:B8,",",TRUE,A2:A8,C1),",","/u"),"")

In C1 type either Fruit or Metal to see that list appear in C1:C8

Paste the following code into a [worksheet module](https://old.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function CELLARRAY(rng As Variant, ParamArray arguments() As Variant)

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'CELLARRAY( range, \*delimiter[s], [optional] "/h", [optional] "/u")

'v1.5 rewrote large parts after fresh revisit - 20190124

'-----------

Dim orientVert As Boolean: orientVert = True ' flag to orient the return array: default is verticle array

Dim arl As Long ' count of elements as array of cells selected

Dim tmpStr As Variant 'build cell contents for conversion to array

Dim str() As String 'the array string

Dim uB As Long: uB = UBound(arguments)

Dim arg As Long, cell As Range, i As Double ', ii As Double

Dim delim As String: delim = "ì" 'will need to be changed if this is your delimiter or character in the data

Dim Unque As Boolean: Unque = False 'return unique data switch

'----generate string of delimited values

If TypeName(rng) = "String" Then 'for string array

tmpStr = rng & delim

Else

For Each cell In rng 'for range

tmpStr = tmpStr + CStr(cell.Value) & delim

Next

End If

'--check for switches for horizontal and unique and convert as required

For arg = 0 To uB

If UCase(arguments(arg)) = "/H" Then

orientVert = False

ElseIf UCase(arguments(arg)) = "/U" Then

Unque = True

Else '--convert delimiters listed to single delimiter for split function

tmpStr = Replace(tmpStr, arguments(arg), delim)

End If

Next

'--remove first and last delimiter at front and end of text if exists

If Left(tmpStr, 1) = delim Then tmpStr = Right(tmpStr, Len(tmpStr) - 1)

If Right(tmpStr, 1) = delim Then tmpStr = Left(tmpStr, Len(tmpStr) - 1)

'------Split the delimited string into an array

str = Split(tmpStr, delim)

'-----get required loop count, for array or cell selection size

arl = Len(tmpStr) - Len(WorksheetFunction.Substitute(tmpStr, delim, ""))

'------------put values into Collection to make unique if /u switch

If Unque Then

Dim coll As Collection

Dim cl As Long

Dim c As Variant

Set coll = New Collection

On Error Resume Next

For i = 0 To arl

c = Trim(str(i))

c = IIf(IsNumeric(c), c \* 1, c) 'load numbers as numbers

coll.Add c, CStr(IIf(Unque, c, i)) 'load unique values if flag is [/U]nique

Next

cl = coll.Count

'--------empty Collection into array for final function return

Dim tempArr() As Variant

ReDim tempArr(cl - 1)

For i = 0 To cl - 1

tempArr(i) = coll.Item(i + 1) 'get the final trimmed element values

Next

CELLARRAY = IIf(orientVert, WorksheetFunction.Transpose(tempArr), tempArr)

Exit Function

End If

'for non unique return the whole array of values

CELLARRAY = IIf(orientVert, WorksheetFunction.Transpose(str), str)

End Function

see also [SPLITIT](https://old.reddit.com/r/excelevator/comments/5j6j9d/udf_splitit_value_delimiter_element_optional_txt/) to return single element values from a list of values in a cell, or the location of a know value in the list of values to help return value pairs[.](https://old.reddit.com/r/excelevator/comments/5izshm/udf_cellarray_range_delimeter_optional_horizontal)

[See SPLITIT and CELLARRAY in use to return an](https://old.reddit.com/r/excel/comments/6jww95/whats_a_good_way_to_parse_this_i_have_300_rows_of/) element from a mutli-delimited cell value

See [RETURNELEMENTS](https://old.reddit.com/r/excelevator/comments/91d9nn/udf_returnelements_text_delimiter_return_elements/) to easily return words in a cells.

See [STRIPELEMENTS](https://old.reddit.com/r/excelevator/comments/91d77j/udf_stripelements_text_delimiter_remove_elements/) to easily strip words from a string of text

See [SUBSTITUTES](https://old.reddit.com/r/excelevator/comments/8vf01o/udf_substitutes_value_find1_replace1_find2/) to replace multiple words in a cell

incentive to start writing this idea [here](https://old.reddit.com/r/excel/comments/5ixg93/i_want_to_add_multiple_values_ie_36_29_52_from_a/)

edit 29/07/2017 add worksheet.trim to remove extra spaces in the data

edit 31/05/2018 remove delimiter if it appears at start and/or end of data string

edit 09/09/2018 fix delimiter removal bug

edit 27/07/2018 tidied up code, numbers now returned as numbers not text

edit 24/01/2019 Rewrite of large portions, tidy up logic and looping

## CRNG - return non-contiguous ranges as contiguous for Excel functions

CRNG( rng1 [ , rng2 , rng3 , ...])

CRNG returns a set of non-contiguous range values as a contiguous range of values allowing the use of non-contiguous ranges in Excel functions.

| **Val1** | **Val2** | **Val3** | **Val4** | **Val5** | **Val6** |
| --- | --- | --- | --- | --- | --- |
| 10 | 20 | - | 30 | - | 40 |

CRNG(A2:B2,D2,F2) returns {10,20,30,40}

Wrap in TRANSPOSE to return a vertical array {10;20;30;40}

| **Function** | **Answer** | **ArrayFormula enter with ctrl+shift+enter** |
| --- | --- | --- |
| Average > 10 | 30 | =AVERAGE(IF(CRNG(A2:B2,D2,F2)>10,CRNG(A2:B2,D2,F2))) |
| Min > 10 | 20 | =MIN(IF(CRNG(A2:B2,D2,F2)>10,CRNG(A2:B2,D2,F2))) |

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function CRNG(ParamArray arguments() As Variant) As Variant

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'CRNG( range1 [, range2, range3....])

Dim uB As Double: uB = UBound(arguments)

Dim str() As Variant, rdp As Long, cell As Range, rcells as long

Dim arr As Long: arr = 0

For rcells = 0 To uB

rdp = rdp + arguments(rcells).Count + IIf(rcells = 0, -1, 0)

ReDim Preserve str(rdp)

For Each cell In arguments(rcells)

str(arr) = cell.Value

arr = arr + 1

Next

Next

CRNG = str()

End Function

## FRNG - return a filtered range of values for IFS functionality in standard functions

FRNG ( total\_rng , criteria\_rng1 , criteria1 [ , criteria\_rng2 , criteria2 , ...])

FRNG returns an array of filtered values from given criteria against a range or ranges. This allows the user to add IFS functionality to some functions that accept ranges as arguments. It should be noted that it does not work with all functions; RANK being one of those - not sure why they do not like array arguments. A bit odd and seemingly random.

| **Values** | **Filter1** | **Filter2** |
| --- | --- | --- |
| 10 | a | x |
| 20 | b | x |
| 30 | a | x |
| 40 | b | x |
| 50 | a | x |
| 60 | b | y |
| 70 | a | y |
| 80 | b | y |
| 90 | a | y |
| 100 | b | y |

| **Filter1** | **Filter2** | **Sum with filtered range (this table at A13)** |
| --- | --- | --- |
| a | x | =SUM( FRNG($A$2:$A$11,$B$2:$B$11,A14,$C$2:$C$11,B14) ) |
| a | x | 90 |
| b | y | 240 |

Yes I know there is SUMIFS, the above is just to show functionality of FRNG and how the filtered range can be used in range arguments.

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function FRNG(rng As Range, ParamArray arguments() As Variant) As Variant

'FRNG ( value\_range , criteria\_range1 , criteria1 , [critera\_range2 , criteria2]...)

'return a filtered array of values for IFS functionality

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim uB As Long, arg As Long, args As Long

Dim i As Long, irc As Long, l As Long, ac As Long

Dim booleanArray() As Boolean, FRNGtr() As Double

On Error Resume Next

i = (rng.Rows.Count \* rng.Columns.Count) - 1

ReDim booleanArray(i)

For l = 0 To i 'initialize array to TRUE

booleanArray(l) = True

Next

uB = UBound(arguments)

args = uB - 1

For arg = 0 To args Step 2 'set the boolean map for matching criteria across all criteria

l = 0

For Each cell In arguments(arg)

If booleanArray(l) = True Then

If TypeName(cell.Value2) = "Double" Then

If TypeName(arguments(arg + 1)) = "String" Then

If Not Evaluate(cell.Value2 & arguments(arg + 1)) Then

booleanArray(l) = False

End If

Else

If Not Evaluate(cell.Value = arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

Else

If Not UCase(cell.Value) Like UCase(arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

If booleanArray(l) = False Then

irc = irc + 1

End If

End If

l = l + 1

Next

Next

ReDim FRNGtr(UBound(booleanArray) - irc) 'initialize array for function arguments

ac = 0

For arg = 0 To i 'use boolean map to build array for stdev

If booleanArray(arg) = True Then

FRNGtr(ac) = rng(arg + 1).Value 'build the value array for MAX

ac = ac + 1

End If

Next

FRNG = FRNGtr()

End Function

## RETURNCOLUMNS - return chosen columns from dataset in any order, with optional limit on rows returned

RETURNCOLUMNS ( [row\_limit] , RANGE , col1 [ , col2 , .. ] )

RETURNCOLUMNS allows you to quickly return an array of columns from a reference data range, any column, any amount of times, simply by referencing the index of the column.

RETURNCOLUMNS allows you to set a row limit on the data returned with the optional first argument as an integer value

This allows for [dynamic use and render of arrays](https://techcommunity.microsoft.com/t5/Excel-Blog/Preview-of-Dynamic-Arrays-in-Excel/ba-p/252944) with the new features coming in Excel 365

Note the Excel VBA array limit of 65536 rows of data applies to this UDF in older versions - just be aware

Following are examples with this as the source data

| **colA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| A21 | B22 | C23 | D24 |
| A31 | B32 | C33 | D34 |
| A41 | B42 | C43 | D44 |
| A51 | B52 | C53 | D54 |
| A61 | B62 | C63 | D64 |
| A71 | B72 | C73 | D74 |
| A81 | B82 | C83 | D84 |
| A91 | B92 | C93 | D94 |
| A101 | B102 | C103 | D104 |

VLOOKUP ColD and return ColB - a right to left lookup.

=VLOOKUP("D54",returncolumns(A1:D10,4,2),2,0) returns B52

Return a reverse columns table

=RETURNCOLUMNS(A1:D10,4,3,2,1) returns the following array

| **ColD** | **ColC** | **ColB** | **colA** |
| --- | --- | --- | --- |
| D24 | C23 | B22 | A21 |
| D34 | C33 | B32 | A31 |
| D44 | C43 | B42 | A41 |
| D54 | C53 | B52 | A51 |
| D64 | C63 | B62 | A61 |
| D74 | C73 | B72 | A71 |
| D84 | C83 | B82 | A81 |
| D94 | C93 | B92 | A91 |
| D104 | C103 | B102 | A101 |

Return columns 3 and 4

=RETURNCOLUMNS(A1:D10,4,3) returns the following array

| **ColD** | **ColC** |
| --- | --- |
| D24 | C23 |
| D34 | C33 |
| D44 | C43 |
| D54 | C53 |
| D64 | C63 |
| D74 | C73 |
| D84 | C83 |
| D94 | C93 |
| D104 | C103 |

Return the first 6 rows of columns 2 and 3

=RETURNCOLUMNS(6,A1:D10,2,3) returns the following array

| **ColB** | **ColC** |
| --- | --- |
| B22 | C23 |
| B32 | C33 |
| B42 | C43 |
| B52 | C53 |
| B62 | C63 |

Return column 1 interspaced between columns 2,3,4

=RETURNCOLUMNS(A1:D4,1,2,1,3,1,4) returns the following array

| **colA** | **ColB** | **colA** | **ColC** | **colA** | **ColD** |
| --- | --- | --- | --- | --- | --- |
| A21 | B22 | A21 | C23 | A21 | D24 |
| A31 | B32 | A31 | C33 | A31 | D34 |
| A41 | B42 | A41 | C43 | A41 | D44 |

Return the first 6 rows of columns 4, 3, 2, 1 and transpose them

=TRANSPOSE(RETURNCOLUMNS(6,A1:D10,4,3,2,1)) returns the following array

| **ColD** | **D24** | **D34** | **D44** | **D54** | **D64** |
| --- | --- | --- | --- | --- | --- |
| ColC | C23 | C33 | C43 | C53 | C63 |
| ColB | B22 | B32 | B42 | B52 | B62 |
| colA | A21 | A31 | A41 | A51 | A61 |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function RETURNCOLUMNS(ParamArray arguments() As Variant) As Variant

'RETURNCOLUMNS ( [row-limit] , RANGE , col1 [ , col2 , .. ] ) : v1.31

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim rtnArray() As Variant

Dim uB As Integer, i As Double, ii As Double, rc As Long, starti As Integer

starti = IIf(TypeName(arguments(0)) = "Double", 1, 0)

uB = UBound(arguments)

If TypeName(arguments(starti)) = "Range" Then

rc = arguments(starti).Rows.Count

Else

rc = UBound(arguments(starti))

End If

rc = IIf(starti, WorksheetFunction.Min(arguments(0), rc), rc)

ReDim rtnArray(rc - 1, uB - 1 - starti)

For i = 0 To uB - 1 - starti

For ii = 0 To rc - 1

rtnArray(ii, i) = arguments(starti)(ii + 1, arguments(i + 1 + starti))

Next

Next

RETURNCOLUMNS = rtnArray()

End Function

## REPTX - Repeat given values to an output array.

REPTX ( textValue , repeat\_x\_times [, return\_horizonal\_array] )

Another function evolved from the new dynamic array paradigm.

Excel has the REPT function that allows the user to repeat given text x times, and little else.

REPTX allows the user to return x number of values to an array.

The textValue can be from a range of cells, a dynamic formula, or another function passing an array.

The repeat\_x\_times is a paired values to repeat that text x times, the argument being from a range or array argument.

By default a vertical array is return by the function. If you wish to return a horizontal array, the third optional boolean argument horizontal should be TRUEor 1

The array will be spilled to the cells with Excel 365.

Examples

REPTX is an array function and returns an array

| **Show** | **Repeat x times** | **String** |
| --- | --- | --- |
| 1 | 2 | Apple |
| 0 | 1 | Banana |
| 1 | 4 | Pear |
| 0 | 3 | Cherry |
| 1 | 5 | Potato |

| **=REPTX(C2:C6,B2:B6)** | **=REPTX(""""&C2:C6&"""",IF(A2:A6,B2:B6))** |
| --- | --- |
| Apple | "Apple" |
| Apple | "Apple" |
| Banana | "Pear" |
| Pear | "Pear" |
| Pear | "Pear" |
| Pear | "Pear" |
| Pear | "Potato" |
| Cherry | "Potato" |
| Cherry | "Potato" |
| Cherry | "Potato" |
| Potato | "Potato" |
| Potato |  |
| Potato |  |
| Potato |  |
| Potato |  |
|  |  |

| **=TEXTJOIN(",",TRUE,REPTX(C2:C6,B2:B6))** |
| --- |
| Apple,Apple,Banana,Pear,Pear,Pear,Pear,Cherry,Cherry,Cherry,Potato,Potato,Potato,Potato,Potato |

| **=REPTX(C2:C6,B2:B6,1)** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Apple | Apple | Banana | Pear | Pear | Pear | Pear | Cherry | Cherry | Cherry | Potato | Potato | Potato | Potato | Potato |

=REPTX({"male","female"},{4,6})

| **List** |
| --- |
| male |
| male |
| female |
| female |
| female |

Paste the following code into a [worksheet module](https://old.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function REPTX(strRng As Variant, repRng As Variant, Optional horizontal As Boolean)

'REPTX ( text , repeat\_x\_times [,return\_horizonal\_array] )

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim rALen As Double 'the length of the arguments

If TypeName(repRng) = "Variant()" Then

rALen = UBound(repRng) - 1

Else

rALen = repRng.Count - 1

End If

Dim rArray()

ReDim rArray(1, rALen) 'the process array

'get the required numner of rows for the final array

Dim ai As Integer: ai = 0

Dim fALen As Double: fALen = 0

Dim fAALen As Integer: fAALen = 0

Dim v As Variant

'& insert the word repeat value to the process array

For Each v In repRng

fALen = fALen + v

rArray(0, ai) = v

ai = ai + 1

fAALen = fAALen + v

Next

Dim fAArray() As Variant 'the final result array

Dim i As Double, ii As Double

ReDim fAArray(fAALen - 1)

'put the words in the process array

i = 0

For Each v In strRng

rArray(1, i) = v

i = i + 1

If i = ai Then Exit For

Next

i = 0

ai = 0

For i = 0 To rALen

For ii = 0 To rArray(0, i) - 1

fAArray(ai) = rArray(1, i)

ai = ai + 1

Next

Next

REPTX = IIf(horizontal, fAArray, WorksheetFunction.Transpose(fAArray))

End Function

## SEQUENCE – Microsoft’s new sequence generator

SEQUENCE emulates Microsoft’s [SEQUENCE](https://support.office.com/en-us/article/sequence-function-57467a98-57e0-4817-9f14-2eb78519ca90) function whereby it generates an array of values as specified by user input.

To create an array of values on the worksheet you can select the area and enter the formula in the active cell with ctrl+shift+enter for the selected cell range to be populated with the array. Alternatively just reference as required in your formula.

ROWS - the row count for the array

COLUMN - an option value for the the column count for the array, the default is 1

Start - an optional value at which to start number sequence, the default is 1

Step - an optional value at which to increment/decrement the values, step default is 1

See [SEQUENCER](https://old.reddit.com/r/excelevator/comments/ao2knk/udf_sequencer_rangecolumns_optional_rows_start/) for sequencing with a vertical value population option and dynamic size specifier from a range.

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function SEQUENCE(nRows As Double, Optional nCols As Variant, Optional nStart As Variant, Optional nStep As Variant) As Variant

'SEQUENCE(rows,[columns],[start],[step])

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

If IsMissing(nCols) Then nCols = 1

If IsMissing(nStart) Then nStart = 1

If IsMissing(nStep) Then nStep = 1

Dim arrayVal() As Variant

ReDim arrayVal(nRows - 1, nCols - 1)

Dim i As Double, ii As Double

For i = 0 To nRows - 1

For ii = 0 To nCols - 1

arrayVal(i, ii) = nStart

nStart = nStart + nStep

Next

Next

SEQUENCE = arrayVal

End Function

## SEQUENCER - sequence with more options, dynamic range match to other range, vertical value population in array

A sequencer UDF - an upgrade to Microsofts SEQUENCE function

SEQUENCER ( range/columns [, rows , start , step , vertical] )

SEQUENCER allows for quick and easy creation of a sequence within an array. The size of the array can be dynamic through reference to a Table or Named range to match the size, or chosen by the user using a constant value or dynamically via a formula.

SEQUENCER has a "v" switch for vertical population of the array value sequence, whereby horizontal population is the result. The "v" switch can be put in place of any argument after the first one, or at the end in its own place. The horizontal switch forces the sequence to be populated vertically rather than horizontally in the array. This is not the same as transposing the array. The array can be transposed by wrapping in the TRANSPOSE function.

To create a grid of a sequence of values, select that range and enter the formula in the active cell and enter with ctrl+shift+enter. If you select a range larger than the array parameters cater for, those array elements will be populated with #N/A

An interesting way to see the formula in action is to select a large range for the function and use 5 reference cells for the arguments, populating those values you will see the array generated dynamically in your selected region.

[See here for example .gif](https://i.imgur.com/yWXAEqx.gif)

Scroll down to the UDF Code after the examples

So many options available, only your imagination is the limit.

4 rows 3 columns - sequence 1 thru 12

=SEQUENCER (4,3)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

4 rows 3 columns, start at 10 thru 21

=SEQUENCER(4,3,10)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 |

4 rows 3 columns, start at 100, step by 15 to 265

=SEQUENCER(4,3,100,15)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 100 | 115 | 130 | 145 |
| 160 | 175 | 190 | 205 |
| 220 | 235 | 250 | 265 |

4 rows 3 columns, step back by -15

=SEQUENCER(4,3,0,-15)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 0 | -15 | -30 | -45 |
| -60 | -75 | -90 | -105 |
| -120 | -135 | -150 | -165 |

Change the direction of the values for a vertical sequence, 4 rows 3 columns start at 10 step 10

=SEQUENCER(4,3,10,10,"v")

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 10 | 40 | 70 | 100 |
| 20 | 50 | 80 | 110 |
| 30 | 60 | 90 | 120 |

Use a range to set the row column values, a Table is a dynamic range and so the array will match those dimensions dynamically

=SEQUENCER(Table1)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

Vertical sequence of dynamic range

=SEQUENCER(Table1,"v")

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |
|  |  | | |

Vertical sequence of dynamic range, start at 10 step 10, vertical values step

=SEQUENCER(Table1,10,10,"v")

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 10 | 40 | 70 | 100 |
| 20 | 50 | 80 | 110 |
| 30 | 60 | 90 | 120 |

A vertical Table of Pi incremented by Pi

=SEQUENCER(Table1,PI(),PI(),"v")

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 3.141593 | 12.56637 | 21.99115 | 31.41593 |
| 6.283185 | 15.70796 | 25.13274 | 34.55752 |
| 9.424778 | 18.84956 | 28.27433 | 37.69911 |

A Table of single values

=SEQUENCER(Table1,10,0)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 |
| 10 | 10 | 10 | 10 |

A Table of the alphabet

=CHAR(SEQUENCER(Table1)+64)

| **ColA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| A | B | C | D |
| E | F | G | H |
| I | J | K | L |

So many uses, this does not even scratch the surface!

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function SEQUENCER(vxAxis As Variant, Optional arg1 As Variant, Optional arg2 As Variant, Optional arg3 As Variant, Optional arg4 As Variant) As Variant

'SEQUENCER ( range , [start] , [step] , [vertical] ) v1.3

'SEQUENCER ( xCount , yCount , [start] , [step] , [vertical] )

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Const vert As String = "v" ' vertical array value path flag

Dim arrayVal() As Variant

Dim xAxis As Double, yAxis As Double

Dim nStart As Double, nStep As Double

Dim uB As Integer, i As Double, ii As Double, iv As Double, isRng As Boolean, orientVert As Boolean

Dim oLoop As Double, iLoop As Double, arRow As Integer, arCol As Integer

If IsMissing(arg1) Then arg1 = ""

If IsMissing(arg2) Then arg2 = ""

If IsMissing(arg3) Then arg3 = ""

If IsMissing(arg4) Then arg4 = ""

Dim goVert As Boolean: goVert = InStr(LCase(arg1 & arg2 & arg3 & arg4), vert)

If TypeName(vxAxis) = "Range" Then

Dim rc As Double: rc = vxAxis.Rows.Count

Dim cc As Double: cc = vxAxis.Columns.Count

If rc \* cc > 1 Then isRng = True

End If

If isRng Then

xAxis = rc

yAxis = cc

If (arg1 = "" Or arg1 = LCase(vert)) Then nStart = 1 Else nStart = arg1

If (arg2 = "" Or arg2 = LCase(vert)) Then nStep = 1 Else nStep = arg2

If (arg3 = "" Or arg3 = LCase(vert)) Then arg2 = 1 Else nStep = arg2

Else

xAxis = IIf(arg1 = "" Or arg1 = LCase(vert), 1, arg1)

yAxis = vxAxis

If (arg2 = "" Or arg2 = LCase(vert)) Then nStart = 1 Else nStart = arg2

If (arg3 = "" Or arg3 = LCase(vert)) Then nStep = 1 Else nStep = arg3

End If

ReDim arrayVal(xAxis - 1, yAxis - 1)

oLoop = IIf(goVert, yAxis - 1, xAxis - 1)

iLoop = IIf(goVert, xAxis - 1, yAxis - 1)

For i = 0 To oLoop

iv = 0

For ii = 0 To iLoop

If goVert Then

arrayVal(iv, i) = nStart

Else

arrayVal(i, ii) = nStart

End If

nStart = nStart + nStep

iv = iv + 1

Next

Next

SEQUENCER = arrayVal

End Function

## SPLITIT - return element value from text array, or array location of text.

Updated to take a RANGE or ARRAY or VALUE as input.

SPLITIT will return a given element within an array of text, or the location of the element containing the text - the array being anything the user determines is splitting the text into elements of an array.

This dual functionality allows for the easy return of paired values within the text array.

Use: =SPLITIT( range , delimiter , return\_element, [optional] txt )

range is a cell, or cells, or array as input

delimiter is whatever you determine that delimits the text array elements, or for an array or range "," is the expected delimiter.

return\_element any argument that returns a number to indicate the required element. This value is ignored when a txt value is entered and is recommended to be 0 where the 'txt' option is used.

txt an optional value - any text to search for in an element of the array for the function to return that array element ID.

Examples

=SPLITIT( A1 , "." , 3 ) returns 100 where A1 = 172.50.100.5

=SPLITIT( A1 , "," , 0 , "Peter" ) returns 2 where A1 = Allen,Peter,age,10

=SPLITIT( A1 , "." , SPLITIT( A1 , "." , 0 , "Allen" )+1 ) returns Peter where A1 = Allen.Peter.age.10

=SPLITIT( "192.168.11.12" , "." , 2 ) returns 168

=SPLITIT( A1:A10 , "," , 3 ) returns the value in A3

=SPLITIT("Sun/Mon/Tue/Wed/Thu/Fri/Sat","/",WEEKDAY(TODAY())) returns the current day of the week

=SPLITIT( CELLARRAY(A1,"/") , "," , 3 ) returns "C" where A1 = A/B/C/D/E

SPLITIT can also be used to extract values from a column mixed with blank cells as it removes blank values by default from the internal array. We use row number to return the values in order.

| **Value list** | **SPLITIT** |
| --- | --- |
| one | =IFERROR(SPLITIT($A$2:$A$12,",",ROW(A1)),"") |
| two | two |
|  | three |
| three | four |
|  | five |
| four |  |
| five |  |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function SPLITIT(rng As Variant, del As String, elmt As Variant, Optional txt As Variant)

'SPLITIT( range , delimiter , return\_element, [optional] txt ) v1.2

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim loopit As Boolean, cell As Range, str As String, i As Double, trimmit As Boolean, relmt As Double

If IsArray(elmt) Then relmt = elmt(1) Else relmt = elmt

If Not IsMissing(txt) Then

loopit = True

End If

If TypeName(rng) = "Variant()" Then

SPLITIT = WorksheetFunction.Transpose(rng)(relmt)

Exit Function

ElseIf TypeName(rng) <> "String" Then

For Each cell In rng

If Trim(cell) <> "" Then str = str & WorksheetFunction.Trim(cell) & del

Next

trimmit = True

Else

str = WorksheetFunction.Trim(rng)

End If

Dim a() As String

a = Split(IIf(trimmit, Left(str, Len(str) - Len(del)), str), del)

If loopit Then

For i = 0 To UBound(a)

If Trim(a(i)) = txt Then

SPLITIT = i + 1

Exit Function

End If

Next

End If

SPLITIT = a(relmt - 1)

End Function

See the [CELLARRAY](https://old.reddit.com/r/excelevator/comments/6f0p2x/udf_cellarray_text_or_range_delimeter_optional/) function to return cell values as an array

[See SPLITIT and CELLARRAY in use to return an](https://old.reddit.com/r/excel/comments/6jww95/whats_a_good_way_to_parse_this_i_have_300_rows_of/) element from a mutli-delimited cell value

## STACKCOLUMNS - stack referenced ranges into columns of your width choice

STACKCOLUMNS ( column\_stack\_width , range1 [ , range2 .. ])

STACKCOLUMNS allows you to stack referenced ranges into a set number of columns in an array.

STACKCOLUMNS takes the referenced non contiguous ranges and stacks them into a contiguous range in an array.

This allows you to format disparate data for querying as a contiguous block of data.

This allows you to combine same table types into a single array; for headers include the whole table for the first reference Table1[#ALL] and just the table body for the tables to stack Table2,Table3,Table4, do not forget the first argument to match the width of the tables.

This allows for [dynamic use and render of arrays](https://techcommunity.microsoft.com/t5/Excel-Blog/Preview-of-Dynamic-Arrays-in-Excel/ba-p/252944) with the new features coming in Excel 365 and should populate to a full table from a single formula in cell. The whole table will then dynamically update with any change made to the source data.

To generate a dynamic array table in current Excel, select a range of cells and enter the formula in the active cell and enter with ctrl+shift+enter for the array to render across the selected cells. Cells outside the array will evaluate to #N/A

column\_stack\_width is the width of the range to be generated and allows for disparate width references to be used to add up to the column\_stack\_width width.

The range arguments are to contain references to ranges to stack across the chosen count of columns.

The function takes each range argument, separates out the columns, and stacks them from left to right. When the last column is filled the next column of data is placed in column 1 below, and then across to fill the column count.

The user must create range references that balance out when stacked. ie. If you have a target of 2 columns, each group of 2 column references should be the same length to balance the stacking. Weird and wonderful results will entail if the ranges to not match to stack correctly.

Note the Excel VBA array limit of 65536 rows of data applies to this UDF in older versions - just be aware

Examples

Stack same type tables sharing attributes and width, In this example the tables are 5 columns wide using the header the first table for the array header row.

=STACKCOLUMNS( 5 , Table1[#All], Table2, Table9, Table25 )

The following are examples with this table as the source data

| **colA** | **ColB** | **ColC** | **ColD** |
| --- | --- | --- | --- |
| A1 | B1 | C1 | D1 |
| A2 | B2 | C2 | D2 |
| A3 | B3 | C3 | D3 |
| A4 | B4 | C4 | D4 |
| A5 | B5 | C5 | D5 |
| A6 | B6 | C6 | D6 |
| A7 | B7 | C7 | D7 |
| A8 | B8 | C8 | D8 |
| A9 | B9 | C9 | D9 |
| A10 | B10 | C10 | D10 |

Stack data from 3 range references, of disparate widths, to 3 columns wide.

=STACKCOLUMNS(3,A1:C5,D6:D11,A6:B11) returns

| **colA** | **ColB** | **ColC** |
| --- | --- | --- |
| A1 | B1 | C1 |
| A2 | B2 | C2 |
| A3 | B3 | C3 |
| A4 | B4 | C4 |
| D5 | A5 | B5 |
| D6 | A6 | B6 |
| D7 | A7 | B7 |
| D8 | A8 | B8 |
| D9 | A9 | B9 |
| D10 | A10 | B10 |

Stack data from 4 range references, to 2 columns wide.

=STACKCOLUMNS(2,A2:D3,C6:D7,A8:D9,A4:B5) returns

| **A1** | **B1** |
| --- | --- |
| A2 | B2 |
| C1 | D1 |
| C2 | D2 |
| C5 | D5 |
| C6 | D6 |
| A7 | B7 |
| A8 | B8 |
| C7 | D7 |
| C8 | D8 |
| A3 | B3 |
| A4 | B4 |

Stack columns from two columns and 8 rows from a Table the [**RETURNCOLUMN**](https://old.reddit.com/r/excelevator/comments/awhi9v/udf_returncolumns_row_limit_range_col1_col2/)'s function that can limit the rows returned of a chosen set of columns or table

=STACKCOLUMNS(2,RETURNCOLUMNS(8,Table1[#All],3,4))

| **ColC** | **ColD** |
| --- | --- |
| C1 | D1 |
| C2 | D2 |
| C3 | D3 |
| C4 | D4 |
| C5 | D5 |
| C6 | D6 |
| C7 | D7 |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function STACKCOLUMNS(grp As Integer, ParamArray arguments() As Variant) As Variant

'STACKCOLUMNS ( group , col1 [ , col2 , .. ] ) v1.31 - take range input for return, limit rows

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim rtnArray() As Variant

Dim uB As Integer, i As Double, ii As Double, j As Double, rRows As Double, rCols As Double

Dim rowPaste As Long: rowPaste = 0 'paste array group index

Dim newPasteRow As Double

Dim colCount As Integer

Dim aRows As Double

uB = UBound(arguments) 'ubound() rows, ubount( ,2) columns, array Variant()

For i = 0 To uB 'get final array size

If TypeName(arguments(i)) = "Variant()" Then

aRows = aRows + (UBound(arguments(i)) / grp \* UBound(arguments(i), 2))

Else

aRows = aRows + (arguments(i).Rows.Count / grp \* arguments(i).Columns.Count)

End If

Next

ReDim Preserve rtnArray(aRows - 1, grp - 1) 'intialise array

'-----------------------------------

'lets get these loops sorted now....

For i = 0 To uB 'need to loop for either array or range

If TypeName(arguments(i)) = "Variant()" Then

rRows = UBound(arguments(i))

rCols = UBound(arguments(i), 2)

Else

rRows = arguments(i).Rows.Count

rCols = arguments(i).Columns.Count

End If

For j = 1 To rCols

colCount = colCount + 1

rowPaste = newPasteRow

'-------------------------

For ii = 1 To rRows

rtnArray(rowPaste, colCount - 1) = arguments(i)(ii, j)

rowPaste = rowPaste + 1

Next

'-------------------------

If colCount = grp Then

colCount = 0

newPasteRow = newPasteRow + rRows

rowPaste = newPasteRow

End If

Next

Next

STACKCOLUMNS = rtnArray()

End Function

## UNPIVOTCOLUMNS - an unpivot function. Unpivot data to an array for use in formulas or output to a table.

UNPIVOTCOLUMNS ( Range , Column\_name , col1/range1 [ , col2/range2 , .. ] )

Data is often recorded and stored in a pivoted style of data across columns for an item. This can make it tricky to create formulas to extract simple answers to data questions.

Office 2016 introduced an UNPIVOT process in PowerQuery to unpivot data to another table.

This UDF unpivots data to an array, allowing the user to use unpivoted data in formulas, or output to the page in an array.

Range - the table of data to unpivot including the header row for the data.

Column\_name - the name to give the new unpivoted column

Col1/range1 - users can refence the columns to unpivot either by an index number of their column position in the table, or as a range of the header cell to unpivot. e.g 2,3,4,6 or B10:B12,B14 or mixed B10:B12,6

The function and result can be used as an argument in a formula to more easily access and query the data.

The function and result can be used to generate a dynamic unpivoted table by selecting a range of cells and entering the formula as an array formula with ctrl+shift+enter.

The function and result can be used to generate a Dynamic Array of an unpivoted table with the new features coming in Excel 365, an instant table of the unpivoted data.

To cement the data, simply copy, paste special values.

Note the Excel VBA array limit of 65536 rows of data applies to this UDF in older versions - just be aware

Examples using this small table of data, which is **Table1** sitting in the range **D25:K28**

| **Company** | **January** | **February** | **March** | **April** | **Region** | **May** | **June** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CompanyA | 1 | 2 | 3 | 4 | RegionA | 5 | 6 |
| CompanyB | 10 | 20 | 30 | 40 | RegionB | 50 | 60 |
| CompanyC | 100 | 200 | 300 | 400 | RegionC | 500 | 600 |

Reference to unpivot a table, with the new column to be labelled **Months** and pivot columns arguments as column indexes 2,3,4,5,7,8

=UNPIVOTCOLUMNS(Table1[#ALL],"Months",2,3,4,5,7,8)

Reference to unpivot a range, with the new column to be labelled **Months** and pivot table column arguments as ranges

=UNPIVOTCOLUMNS(D25:K28,"Months",E25:H25, J25,K25)

Reference to unpivot a Table with the new column to be label taken from cell **A1** and pivot column arguments as Table reference and index combined

=UNPIVOTCOLUMNS(Table1[#All],A1,Table1[[#Headers],[January]:[April]],7,8)

The resulting array;

| **Company** | **Region** | **Months** | **Value** |
| --- | --- | --- | --- |
| CompanyA | RegionA | January | 1 |
| CompanyA | RegionA | February | 2 |
| CompanyA | RegionA | March | 3 |
| CompanyA | RegionA | April | 4 |
| CompanyA | RegionA | May | 5 |
| CompanyA | RegionA | June | 6 |
| CompanyB | RegionB | January | 10 |
| CompanyB | RegionB | February | 20 |
| CompanyB | RegionB | March | 30 |
| CompanyB | RegionB | April | 40 |
| CompanyB | RegionB | May | 50 |
| CompanyB | RegionB | June | 60 |
| CompanyC | RegionC | January | 100 |
| CompanyC | RegionC | February | 200 |
| CompanyC | RegionC | March | 300 |
| CompanyC | RegionC | April | 400 |
| CompanyC | RegionC | May | 500 |
| CompanyC | RegionC | June | 600 |

Use with [**RETURNCOLUMS UDF**](https://old.reddit.com/r/excelevator/comments/awhi9v/udf_returncolumns_row_limit_range_col1_col2/) to return only the second and third columns

=RETURNCOLUMS(UNPIVOTCOLUMNS(Table1[#All],"Month",Table4[[#Headers],[January]:[April]],J25:K25),2,3)

Reference to unpviot the sales months in a table. By only referencing the sales column and returning those rows, we get a table of sales.

=UNPIVOTCOLUMNS(E25:H28,"Sales",1,2,3,4)

| **Sales** | **Value** |
| --- | --- |
| January | 1 |
| February | 2 |
| March | 3 |
| April | 4 |
| January | 10 |
| February | 20 |
| March | 30 |
| April | 40 |
| January | 100 |
| February | 200 |
| March | 300 |
| April | 400 |

Paste the following code into a [worksheet module](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for it to be available for use.

Function UNPIVOTCOLUMNS(rng As Range, cName As Variant, ParamArray arguments() As Variant) As Variant

'UNPIVOTCOLUMNS ( range , colName , col1/range1 [ , col2/range2 , .. ] )

'v2.13 take range arguments for all arguments, allow all columns to unpivot

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim rtnArray() As Variant

Dim i As Double, j As Double, uB As Integer: uB = -1

Dim colCount As Integer: colCount = rng.Columns.Count

Dim rowCount As Double: rowCount = rng.Rows.Count

Dim unpivotedColumnsCount As Integer

Dim newrowcount As Double

Dim printColumns As String

Dim pivotColumns As String

Dim printColsArray() As String

Dim pivotColsArray() As String

Dim lastElement As Integer

For i = 0 To UBound(arguments) 'get the columns to unpivot

If TypeName(arguments(i)) = "Range" Then

For Each cell In arguments(i).Columns

pivotColumns = pivotColumns & (cell.Column - (rng.Cells(1, 1).Column - 1)) & "|"

uB = uB + 1

Next

Else

pivotColumns = pivotColumns & arguments(i) & "|"

uB = uB + 1

End If

Next

pivotColsArray = Split(Left(pivotColumns, Len(pivotColumns) - 1), "|")

headerColumnsCounts = colCount - (uB + 2)

unpivotedColumnsCount = uB - uB + 2

newrowcount = (rowCount) + (rowCount - 1) \* uB

lastElement = headerColumnsCounts + unpivotedColumnsCount

ReDim Preserve rtnArray(newrowcount - 1, lastElement) 'intialise return array

'build array header and get column population index for unpivot

Dim pi As Integer: pi = 0 'param array argument index

Dim aH As Integer: aH = 0 'new array header index

rtnArray(0, lastElement - 1) = cName

rtnArray(0, lastElement) = "Value"

For j = 1 To colCount 'get the header row populated

If j <> pivotColsArray(WorksheetFunction.Min(pi, uB)) Then

rtnArray(0, aH) = rng.Cells(1, j)

aH = aH + 1

printColumns = printColumns & j & "|"

Else

pi = pi + 1

End If

Next

'--------------------end header build

'---get columns index to print and process

If printColumns <> "" Then

printColsArray = Split(Left(printColumns, Len(printColumns) - 1), "|")

'-----------------------------------

'------loop generate the non-pivot duplicate values in the rows

Dim r As Integer, c As Integer, irow As Double: c = 0 'row and column counters

For Each printcolumn In printColsArray 'loop through columns

r = 1 'populate array row

For irow = 2 To rowCount 'loop through source rows

For x = 0 To uB

rtnArray(r, c) = rng.Cells(irow, --printcolumn)

r = r + 1

Next

Next

c = c + 1

Next

End If

'-----------------------------------

'------loop generate the unpivot values in the rows

r = 1: c = 0

For cell = 1 To newrowcount - 1

rtnArray(cell, lastElement - 1) = rng.Cells(1, --pivotColsArray(c)).Value

rtnArray(cell, lastElement) = rng.Cells(r + 1, --pivotColsArray(c)).Value

If c = uB Then c = 0: r = r + 1 Else c = c + 1

Next

UNPIVOTCOLUMNS = rtnArray()

End Function

## VRNG - return array of columns from range as a single array

VRNG ( rng1 [ , rng2 , rng3 , ...])

When given a range of cells Excel evaluates the range on a row by row basis and not on a column by column basis.

VRNG will return an array of column values from a given range in a single vertical array.

This will allow for the processing of a table of cells as a single column in an array

| **Col1** | **Col2** |  | **col3** |
| --- | --- | --- | --- |
| 1 | 4 |  | 7 |
| 2 | 5 |  | 8 |
| 3 | 6 |  | 9 |
| =vrng(A2:B4,D2:D4) |  |  |  |

**Returns** {1;2;3;4;5;6;7;8;9}

If you need the array in horizonal format then wrap in TRANSPOSE for {1,2,3,4,5,6,7,8,9}

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function VRNG(ParamArray arguments() As Variant) As Variant

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim uB As Integer: uB = UBound(arguments)

Dim str() As Variant

Dim cell As Range, column As Range

Dim arg As Integer, i As Double: i = 0

Dim cCount As Double: cCount = -1

For arg = 0 To uB

cCount = cCount + arguments(arg).Count

ReDim Preserve str(cCount)

For Each column In arguments(arg).Columns

For Each cell In column.Cells

str(i) = cell.Value

i = i + 1

Next

Next

Next

VRNG = WorksheetFunction.Transpose(str())

End Function

# IF Functions

## FUNCIFS - IFS criteria for all suitable functions!

FUNCIFS ( "function" , range , criteria\_range1 , criteria1 [ , criteria\_range2 , criteria2 .. ])

FUNCIFS ( "STDEV" , A1:A500 , B1:B100 , "criteria1" [ , criteria\_range2 , criteria2 .. ])

There are a few functions in Excel that could do with having an ..IFS equivalent to SUMIFS, AVERAGEIFS etc.

This DIY UDF allows you to add the required function that you want to be able to filter the value set for, essentially adding ..IFS functionality to any function that takes a range or ranges of cells as input for filtering.

**To add a function**, scroll to the bottom of the function and add another CASE statement with that function. Then simply type that function name in as the first argument.

As an example, the code below has 2 case statments, one for SUM and another for STDEV meaning those two functions now have IFS functionality. Yes I know there exists SUMFIS , it is here for an example.

| **Value** | **filter1** | **filter2** |
| --- | --- | --- |
| 104 | x | o |
| 26 | x |  |
| 756 |  |  |
| 127 | x | o |
| 584 | x | o |
| 768 |  | o |
| 715 | x |  |
| 114 | x | o |
| 381 |  |  |

| **Value** | **Formula** |
| --- | --- |
| 3575 | =FUNCIFS("sum",A2:A10) |
| 1670 | =FUNCIFS("sum",A2:A10,B2:B10,"x") |
| 292.6025746 | =FUNCIFS ("stdev",$A$2:$A$10,B2:B10,"x") |
| 234.6889786 | =FUNCIFS ("stdev",$A$2:$A$10,B2:B10,"x",C2:C10,"o") |

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Then add your function that you want ..IFS filtering for at the end in a new CASE statement.

Function FUNCIFS(func As String, rng As Range, ParamArray arguments() As Variant) As Double

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

'FUNCIFS ( "function" , value\_range , criteria\_range1 , criteria1 , [critera\_range2 , criteria2]...)

Dim uB As Long, arg As Long, args As Long, i As Long, l As Long, irc As Long 'include row count to initialize arrya

Dim booleanArray() As Boolean

Dim valueArray() As Double

i = rng.Count - 1

ReDim booleanArray(i)

For l = 0 To i 'initialize array to TRUE

booleanArray(l) = True

Next

uB = UBound(arguments)

args = uB - 1

For arg = 0 To args Step 2 'set the boolean map for matching criteria across all criteria

l = 0

For Each cell In arguments(arg)

If booleanArray(l) = True Then

If TypeName(cell.Value2) = "Double" Then

If TypeName(arguments(arg + 1)) = "String" Then

If Not Evaluate(cell.Value2 & arguments(arg + 1)) Then

booleanArray(l) = False

End If

Else

If Not Evaluate(cell.Value = arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

Else

If Not UCase(cell.Value) Like UCase(arguments(arg + 1)) Then

booleanArray(l) = False

End If

End If

If booleanArray(l) = False Then

irc = irc + 1

End If

End If

l = l + 1

Next

Next

ReDim valueArray(UBound(booleanArray) - irc) 'initialize array for function arguments

ac = 0

For arg = 0 To i 'use boolean map to build array for stdev

If booleanArray(arg) = True Then

valueArray(ac) = rng(arg + 1).Value 'build the value array for STDEV

ac = ac + 1

End If

Next

Select Case func 'add functions as required here

Case "sum": FUNCIFS = WorksheetFunction.Sum(valueArray)

Case "stdev": FUNCIFS = WorksheetFunction.StDev(valueArray)

'Case "NAME HERE": FUNCIFS = WorksheetFunction.NAME HERE(valueArray) '<==Copy, Edit, Uncomment

'where NAME HERE is the function to call

End Select

End Function

## IFEQUAL - returns expected result when formula returns expected result.

This function returns the expected result when the formula return value matches the expected result, otherwise it returns a user specified value or 0.

It removes the necessity to duplicate long VLOOKUP or INDEX MATCH formulas when a match is being verified.

Use =IFEQUAL ( Value , expected\_result , [Optional] else\_return)

Examples;

=IFEQUAL(A1, 20 ) 'returns 20 if A1 = 20, else returns 0

=IFEQUAL(A1+A2, 20,"wrong answer" ) ' returns 20 if A1+A2 = 20, else returns `wrong answer`

=IFEQUAL(A1+A2, B1+B2, "No") 'returns B1+B2 if A1+A2 = B1+B2, , else returns `No`

=IFEQUAL(A1, ">10" , A2 ) 'returns the value of A2 if A1 is less than 10, else return A1

=IFEQUAL( formula , "<>0" , "" ) 'returns the value of formula if not 0 else return blank

=IFEQUAL( formula , ">0" , "Re order" ) 'returns the value of formula if great than 0 or `Re-order`

=IFEQUAL( formula , "Red" , "Emergency" ) 'returns the value of formula if not `Red` or `Emergency`

[Follow these instructions](https://www.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/) for making the UDF available, using the code below.

Function IFEQUAL(arg As Variant, ans As Variant, Optional neg As Variant)

'IFEQUAL ( formula, expected\_result , optional otherwise ) :V2.5

'https://www.reddit.com/u/excelevator

'https://old.reddit.com/r/excelevator

'https://www.reddit.com/r/excel - for all your Spreadsheet questions!

Dim a As Variant: a = arg

Dim b As Variant: b = ans

Dim c As Variant: c = neg

Dim comp As Boolean: comp = InStr(1, "<>=", Left(b, 1))

Dim eq As Integer: eq = InStr(1, "<>", Left(b, 2)) \* 2

If TypeName(a) = "Double" And \_

TypeName(b) = "String" And comp Then

IFEQUAL = IIf(Evaluate(a & b), a, c)

Exit Function

ElseIf TypeName(a) = "String" And \_

TypeName(b) = "String" And \_

(comp Or eq) Then

IFEQUAL = IIf(Evaluate("""" & a & """" & Left(b, WorksheetFunction.Max(comp, eq)) & """" & Right(b, Len(b) - WorksheetFunction.Max(comp, eq)) & """"), a, c)

Exit Function

End If

IFEQUAL = IIf(a = b, a, c)

End Function

# Appendix A – Links to various solutions on Reddit

## General info

[7 new Excel 365 functions as UDFs for compatibility](https://old.reddit.com/r/excelevator/comments/9jn6p4/6_new_excel_365_functions_as_udfs_for/)

[Arrays and Excel and SUMPRODUCT](https://old.reddit.com/r/excelevator/comments/8xblyy/arrays_and_excel_and_sumproduct/)

[Find first and last day of week](https://old.reddit.com/r/excelevator/comments/4vlflt/find_first_and_last_day_of_week/)

[INDEX ( MATCH ( ) ) How to!](https://old.reddit.com/r/excelevator/comments/3p6bdk/index_match_how_to/)

[Move cursor around data super fast without a mouse](https://old.reddit.com/r/excelevator/comments/4pmf7f/move_cursor_around_data_super_fast_without_a_mouse/)

[Multiple Range use for single range function](https://old.reddit.com/r/excelevator/comments/30nb9o/multiple_range_use_for_single_range_function/)

[Text (formatted date) to Columns to Date](https://old.reddit.com/r/excelevator/comments/8l65rd/text_formatted_date_to_columns_to_date/)

[UDF Locations instructions - Module and Add-Ins](https://old.reddit.com/r/excelevator/comments/2wtdvz/udf_locations_instructions_module_and_addins/)

[Using Command prompt and Excel to get files listing hyperlinked](https://old.reddit.com/r/excelevator/comments/34563c/using_command_prompt_and_excel_to_get_files/)

[Volatile user defined functions](https://old.reddit.com/r/excelevator/comments/35ammc/udf_volatile_functions/)

[Solution list link to questions](https://old.reddit.com/r/excelevator/comments/5fna1z/self_note_for_code_posts/)

## User defined functions

### 365 Functions and similar

[CONCAT](https://old.reddit.com/r/excelevator/comments/8w7b5p/udf_concat_textrange1_textrange2_concatenate/) - concatenate string and ranges

[COUNTUNIQUE](https://old.reddit.com/r/excelevator/comments/efuykv/udf_countunique_value1_value2_get_the_count_of/) get the count of unique values from cells, ranges, arrays, formula results.

[DAYS](https://old.reddit.com/r/excelevator/comments/2vd9ed/excel_days_funtion_for_pre_2013_excel/) - Excel DAYS() funtion for pre 2013 Excel

[FORMULATEXT](https://old.reddit.com/r/excelevator/comments/as5rj2/udf_formulatext_range_return_the_absolute_value/) - return the absolute value of a cell

[IFS](https://old.reddit.com/r/excelevator/comments/5ero0h/udf_ifs_for_pre_3652016_excel/) - return a value if argument is true

[IFVALUES](https://old.reddit.com/r/excelevator/comments/5gm50v/udf_ifvalues_arg_if_value_this_value_if_value/) - returns a given value if the argument is equal to a given value

[ISHYPERLINK](https://old.reddit.com/r/excelevator/comments/3accoc/udf_test_cell_for_hyperlink_ishyperlink/) - test cell for Hyperlink

[ISVISIBLE](https://old.reddit.com/r/excelevator/comments/amxiyd/udf_isvisible_range_optional_hidden_a_visible_or/) - a visible or hidden row mask array - include only hidden or visible rows in calculations

[MAXIFS](https://old.reddit.com/r/excelevator/comments/8vfoag/udf_maxifs_min_range_criteria_range1_criteria1/) - filter the maximum value from a range of values

[MINIFS](https://old.reddit.com/r/excelevator/comments/8vfncs/udf_minifs_min_range_criteria_range1_criteria1/) - filter the minimum value from a range of values

[SWITCH](https://old.reddit.com/r/excelevator/comments/8mwxp2/udf_switch_value_match1_return1_matchx_returnx/) - evaluates one value against a list of values and returns the result corresponding to the first matching value.

[TEXTJOIN](https://old.reddit.com/r/excelevator/comments/5movbv/udf_textjoin_delimeter_ignore_blanks_valuerange/) - combines the text from multiple ranges and/or strings, and includes a delimiter you specify

[TXLOOKUP](https://old.reddit.com/r/excelevator/comments/dbd7n9/udf_txlookup_value_table_lookup_col_return_cols/) - XLOOKUP for Tables/ranges using column names for dynamic column referencing

[UNIQUE](https://old.reddit.com/r/excelevator/comments/8w1ko7/udf_unique_range_optional_count_return_an_array/) - return an array of unique values, or a count of unique values

[XLOOKUP](https://old.reddit.com/r/excelevator/comments/d1rv2b/udf_xlookup_value_lookup_range_return_range_match/) - the poor mans version of the Microsoft XLOOKUP function for Excel 365

### Array functions

[ARRAYIFS](https://old.reddit.com/r/excelevator/comments/batgfm/udf_arrayifs_function_data_column_array_col1_arg1/) - IFS functionality for arrays

[ASG](https://old.reddit.com/r/excelevator/comments/ai9hq8/udf_asg_startnum_endnum_optional_step_array/) - array Sequence Generator - generate custom sequence arrays with ease

[CELLARRAY](https://old.reddit.com/r/excelevator/comments/6f0p2x/udf_cellarray_text_or_range_delimeter_optional/) - return multi delimited cell(s) values as array, switch for horizontal array and/or return unique values

[CRNG](https://old.reddit.com/r/excelevator/comments/9eyz27/udf_crng_rng1_rng2_rng3_return_noncontiguous/) - return non-contiguous ranges as contiguous for Excel functions

[FRNG](https://old.reddit.com/r/excelevator/comments/a15vhg/udf_frng_total_rng_criteria_rng1_criteria1/) - return a filtered range of values for IFS functionality in standard functions

[RETURNCOLUMNS](https://old.reddit.com/r/excelevator/comments/awhi9v/udf_returncolumns_row_limit_range_col1_col2/) - return chosen columns from dataset in any order, with optional limit on rows returned

[REPTX](https://old.reddit.com/r/excelevator/comments/iwzjy3/udf_reptx_text_rangearray_repeat_rangearray/) - Repeat given values to an output array.

[SEQUENCE](https://old.reddit.com/r/excelevator/comments/ao2ip4/udf_sequence_rows_column_start_step_generate_a/) - Microsofts new sequence generator

[SEQUENCER](https://old.reddit.com/r/excelevator/comments/ao2knk/udf_sequencer_rangecolumns_optional_rows_start/) - sequence with more options, dynamic range match to other range, vertical value population in array

[SPLITIT](https://old.reddit.com/r/excelevator/comments/5j6j9d/udf_splitit_value_delimiter_element_optional_txt/) - return element value from text array, or array location of text.

[STACKCOLUMNS](https://old.reddit.com/r/excelevator/comments/b01vxx/udf_stackcolumns_column_stack_width_range1_range2/) - stack referenced ranges into columns of your width choice

[UNPIVOTCOLUMNS](https://old.reddit.com/r/excelevator/comments/b0n4v0/udf_unpivotcolumns_range_column_name_col1range1/) - an unpivot function. Unpivot data to an array for use in formulas or output to a table.

[VRNG](https://old.reddit.com/r/excelevator/comments/aa5u3m/udf_vrng_rng1_rng2_rng3_return_array_of_columns/) - return array of columns from range as a single array

### IF functions

[FUNCIFS](https://old.reddit.com/r/excelevator/comments/6eaxgj/udf_funcifs_function_function_range_criteria/) - IFS criteria for all suitable functions!

[IFEQUAL](https://old.reddit.com/r/excelevator/comments/5et7o1/udf_ifequal_formula_expected_result_optional_else/) - returns expected result when formula returns expected result.

[IFXRETURN](https://old.reddit.com/r/excelevator/comments/8mxxar/udf_ifxreturn_value_match1_rtn1_matchx_rtnx/) - return value when match is not found

[LARGEIFS](https://old.reddit.com/r/excelevator/comments/c2eprz/udf_largeifs_range_large_index_criteria_range1/) - LARGE with IFS criteria

[PERCENTAGEIFS](https://old.reddit.com/r/excelevator/comments/8uiubn/udf_percentageifs_criteria_range1_criteria1/) - return the percentage of values matching multiple criteria

[SMALLIFS](https://old.reddit.com/r/excelevator/comments/c2epct/udf_smallifs_range_small_index_criteria_range1/) - SMALL with IFS criteria

[STDEVIFS](https://old.reddit.com/r/excelevator/comments/659iwl/udf_stdevifs_stdev_range_criteria_range1/) - STDEV with IFS criteria

[SUBTOTALIFS](https://old.reddit.com/r/excelevator/comments/9dia6c/udf_subtotalifs_function_function_range_criteria/) - SUBTOTAL with IFS criteria

[TEXTIFS](https://old.reddit.com/r/excelevator/comments/5rfriw/udf_textifs_return_range_delimiter_ignore_blanks/) - return text against column criteria

### Lookup functions

[ILOOKUP](https://old.reddit.com/r/excelevator/comments/g70nne/udf_ilookup_lookup_value_parentcol_childcol/) - return an array of the iterations of lookup values from parent to child records

[NMATCH](https://old.reddit.com/r/excelevator/comments/arxwh4/udf_nmatch_value_range_instance_optional/) - return the index of the Nth instance of a lookup value

[NMATCHIFS](https://old.reddit.com/r/excelevator/comments/as5ws8/udf_nmatchifs_range_instance_closest_match/) return the index of the Nth match in a column range against multiple criteria

[NVLOOKUP](https://old.reddit.com/r/excelevator/comments/arxwja/udf_nvlookup_value_range_column_instance_optional/) - return the Nth instance of a lookup values associated row column value

[NVLOOKUPIFS](https://old.reddit.com/r/excelevator/comments/as5wt0/udf_nvlookupifs_lookup_value_range_return_col_rtn/) - return the Nth matching record in a row column range against multiple criteria

### Text return and formatting functions

[COMPARETEXT](https://old.reddit.com/r/excelevator/comments/8oplle/udf_comparetext_value1_value2_optional_case/) - text compare with text exclusions and case sensitivity option.

[DELIMSTR](https://old.reddit.com/r/excelevator/comments/acpezs/udf_delimstr_value_delimiter_interval_delimit_a/) - delimit a string with chosen character/s at a chosen interval

[GETCFINFO](https://old.reddit.com/r/excelevator/comments/e1qmow/udf_getcfinfo_range_hascf_0/) - get details of Conditional formatting in a cell or range

[GETDATE](https://old.reddit.com/r/excelevator/comments/erttz1/udf_getdate_range_mask_optional_century_extract/) - Extract the date from text in a cell from a given extraction mask and return the date serial

[GETSTRINGS](https://old.reddit.com/r/excelevator/comments/dfw656/udf_getstrings_source_range_optional_offset/) - Return strings from a cell or range of cells, determined by 1 or multiple filters

[INSERTSTR](https://old.reddit.com/r/excelevator/comments/hsw5mp/udf_insertstr_values_positions1_text1_positionsx/?) - - quickly insert multiple values into existing values - single, multiple, arrays...

[INTXT](https://old.reddit.com/r/excelevator/comments/av6ssu/udf_intxt_rngvalue1_rngvalue2_optional_position/) - return value match result, single, multiple, array, boolean or position

[ISVALUEMASK](https://old.reddit.com/r/excelevator/comments/c90wdo/udf_isvaluemask_value_mask_test_for_a_value/) - test for a value format - return a boolean value against a mask match on a single cell or array of values.

[LDATE](https://old.reddit.com/r/excelevator/comments/i3gbxx/udf_ldate_date_values_delimiter_quickly_convert_a/) - - quickly convert a date to your date locale

[MIDSTRINGX](https://old.reddit.com/r/excelevator/comments/8th17o/udf_midstringx_string_start_end_instance_optional/) - extract instance of repeat string in a string

[MULTIFIND](https://old.reddit.com/r/excelevator/comments/iska2o/udf_multifind_find_range_words_range_case_match/) - return a string/s from multiple search words

[RETURNELEMENTS](https://old.reddit.com/r/excelevator/comments/91d9nn/udf_returnelements_text_delimiter_return_elements/) - quickly return multiple isolated text items from string of text

[STRIPELEMENTS](https://old.reddit.com/r/excelevator/comments/91d77j/udf_stripelements_text_delimiter_remove_elements/) - quickly remove multiple text items from string of text

[SUBSTITUTES](https://old.reddit.com/r/excelevator/comments/8vf01o/udf_substitutes_value_find1_replace1_find2/) - replace multiple values in one formula, no more nested SUBSTITUTE monsters...

[TEXTMASK](https://old.reddit.com/r/excelevator/comments/92mfuh/udf_textmask_range_mask_showchar_hidechar/) - quickly return edited extracted string

[UDF and MACRO - YYYMMDD to dd/mm/yyyy - ISO8601 date format to Excel formatted date](https://old.reddit.com/r/excelevator/comments/65bcj3/udf_and_macro_yyymmdd_to_ddmmyyyy_iso8601_date/)

### Timesheet functions

[TIMECARD](https://old.reddit.com/r/excelevator/comments/caygkf/udf_timecard_start_time_end_time_start_time_end/) - a timesheet function to sum the time between start-end times

[WORKTIME](https://old.reddit.com/r/excelevator/comments/cf7d1p/udf_worktime_datetimestart_datetimeend_work_start/) - sum working hours between 2 dates between given start and end time in those days

### Conditional functions

[ADDVISIBLEONLY](https://old.reddit.com/r/excelevator/comments/351cma/udf_sum_of_cells_on_multiple_sheets_but_only_if/) - sum of Cells on multiple sheets but only if sheets are visible.

[AVERAGE3DIF](https://old.reddit.com/r/excelevator/comments/4wetxp/udf_3d_averageif/) - average across multiple sheets

[SUMBYCOLOUR](https://old.reddit.com/r/excelevator/comments/5v0g1p/udf_sumbycolour_rng_colour_sum_range/) - sum values based on cell colour - does not work for conditional format

[SUPERLOOKUP](https://old.reddit.com/r/excelevator/comments/5e17j0/udf_superlookup_get_information_on_search_result/) - get information on search result cell from a range

[TOPX](https://old.reddit.com/r/excelevator/comments/2wn513/return_top_nth_result_across_a_range_of_cells/) - Return TOP N'th result across a range of cells.

[TOPXA](https://old.reddit.com/r/excelevator/comments/2wn455/return_average_of_x_results_in_a_range/) - Return average of X results in a range

## VBA solutions

[Add/subtract cell value from entry in another cell](https://old.reddit.com/r/excelevator/comments/5b7whz/vba_macro_addsubtract_cell_value_from_entry_in/)

[Complete missing values in list](https://old.reddit.com/r/excelevator/comments/534d1b/vba_macro_complete_missing_values_in_list/)

[Create dynamically named Worksheet](https://old.reddit.com/r/excelevator/comments/36v7ap/vba_create_dynamically_named_worksheet/)

[Do something on cell selection within a range](https://old.reddit.com/r/excelevator/comments/2zse3c/do_something_on_cell_selection_within_a_range/)

[Do something on cell value change within a range](https://old.reddit.com/r/excelevator/comments/3a48r0/do_something_on_cell_value_change_within_a_range/)

[Dynamic List drop down validation from Range](https://old.reddit.com/r/excelevator/comments/2wdri4/dynamic_list_drop_down_validation_from_range/)

[Excel Audit Timestamp](https://old.reddit.com/r/excelevator/comments/35fxnl/excel_audit_timestamp/)

[Excel List validation from cell selection](https://old.reddit.com/r/excelevator/comments/2vtzkm/excel_list_validation_from_cell_selection/)

[Fill column with COUNTIF from previous column over](https://old.reddit.com/r/excelevator/comments/3eibng/macro_fill_column_with_countif_from_previous/)

[Format character/word in a cell](https://old.reddit.com/r/excelevator/comments/5fm5io/vba_macro_format_characterword_in_a_cell/)

[Generate Reddit Table markup from selected region](https://old.reddit.com/r/excelevator/comments/a0fczs/vba_generate_reddit_table_markup_from_selected/)

[How to run a sub routine in Excel](https://old.reddit.com/r/excelevator/comments/a0f82h/vba_how_to_run_a_sub_routine_in_excel/)

[Import CSV and specify column data types](https://old.reddit.com/r/excelevator/comments/5ncbz1/vba_macro_import_csv_and_specify_column_data_types/)

[Pad cells with zer0s](https://old.reddit.com/r/excelevator/comments/5avyuu/vba_macro_pad_cells_with_zer0s/)

[Paste Append data into cell](https://old.reddit.com/r/excelevator/comments/2wqf3w/paste_append_data_into_cell/)

[Pasting data to the end of a column or row](https://old.reddit.com/r/excelevator/comments/2x1mp8/vba_pasting_data_to_the_end_of_a_column_or_row/)

[Plotter - show the path of a plot in a grid from list of cell addresses](https://old.reddit.com/r/excelevator/comments/3a939d/plotter_show_the_path_of_a_plot_in_a_grid_from/)

[Replace values in cells from list of words](https://old.reddit.com/r/excelevator/comments/5ftnha/vba_macro_replace_values_in_cells_from_list_of/)

[Spell check words in selected list](https://old.reddit.com/r/excelevator/comments/3all41/vba_spell_check_words_in_selected_list/)

[Update and Refresh all Pivot tables in a workbook.](https://old.reddit.com/r/excelevator/comments/2x26sr/update_and_refresh_all_pivot_tables_in_a_workbook/)

[**UNPIVOT Data - multi column headers and/or record groups**](https://old.reddit.com/r/excelevator/comments/af8f8l/vba_macro_unpivot_data_multi_column_headers_andor/)

[Write Random numerical values to a range of cells](https://old.reddit.com/r/excelevator/comments/5gbhny/vba_macro_write_random_numerical_values_to_a/)

Short link to this page <https://bit.ly/2JSM1M1>

# Appendix B – Misc. Notes that apply to all UDFs

* Include the following second line of code in your UDF, it makes them volatile, i.e they recalc with every edit made to the worksheet.

Function myfunc( )

Application.Volatile

* Put all your [favorite UDFs](https://old.reddit.com/r/excelevator/) in an [add-in for always there use](https://support.office.com/en-us/article/add-or-remove-add-ins-in-excel-0af570c4-5cf3-4fa9-9b88-403625a0b460)
* For local PC only, insert a module into your current workbook and paste the UDF into the module. Open VBA editor (Alt+F11) > Insert > Module

# Appendix C – non-VBA/UDF tips and tricks

## Acronyms, initialisms, abbreviations, contractions, and other phrases which expand to something larger:

| **Fewer Letters** | **More Letters** |
| --- | --- |
| [AND](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnfb3li) | [Returns TRUE if all of its arguments are TRUE](https://support.microsoft.com/en-us/office/and-function-5f19b2e8-e1df-4408-897a-ce285a19e9d9) |
| [CHOOSE](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnejpjg) | [Chooses a value from a list of values](https://support.microsoft.com/en-us/office/choose-function-fc5c184f-cb62-4ec7-a46e-38653b98f5bc) |
| [COS](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnffhk1) | [Returns the cosine of a number](https://support.microsoft.com/en-us/office/cos-function-0fb808a5-95d6-4553-8148-22aebdce5f05) |
| [IFERROR](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnfikel) | [Returns a value you specify if a formula evaluates to an error; otherwise, returns the result of the formula](https://support.microsoft.com/en-us/office/iferror-function-c526fd07-caeb-47b8-8bb6-63f3e417f611) |
| [RADIANS](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnffhk1) | [Converts degrees to radians](https://support.microsoft.com/en-us/office/radians-function-ac409508-3d48-45f5-ac02-1497c92de5bf) |
| [SIN](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnffhk1) | [Returns the sine of the given angle](https://support.microsoft.com/en-us/office/sin-function-cf0e3432-8b9e-483c-bc55-a76651c95602) |
| [SUMPRODUCT](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnffh44) | [Returns the sum of the products of corresponding array components](https://support.microsoft.com/en-us/office/sumproduct-function-16753e75-9f68-4874-94ac-4d2145a2fd2e) |
| [SWITCH](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnfp9cw) | [Excel 2016+: Evaluates an expression against a list of values and returns the result corresponding to the first matching value. If there is no match, an optional default value may be returned.](https://support.microsoft.com/en-us/office/switch-function-47ab33c0-28ce-4530-8a45-d532ec4aa25e) |
| [UNICHAR](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hngkvic) | [Excel 2013+: Returns the Unicode character that is references by the given numeric value](https://support.microsoft.com/en-us/office/unichar-function-ffeb64f5-f131-44c6-b332-5cd72f0659b8) |
| [VLOOKUP](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnfikel) | [Looks in the first column of an array and moves across the row to return the value of a cell](https://support.microsoft.com/en-us/office/vlookup-function-0bbc8083-26fe-4963-8ab8-93a18ad188a1) |
| [WEBSERVICE](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnfom22) | [Excel 2013+: Returns data from a web service.](https://support.microsoft.com/en-us/office/webservice-function-0546a35a-ecc6-4739-aed7-c0b7ce1562c4) |
| [XLOOKUP](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnejpjg) | [Office 365+: Searches a range or an array, and returns an item corresponding to the first match it finds. If a match doesn't exist, then XLOOKUP can return the closest (approximate) match.](https://support.microsoft.com/en-us/office/xlookup-function-b7fd680e-6d10-43e6-84f9-88eae8bf5929) |
| [XMATCH](https://old.reddit.com/r/Excel/comments/r9ux4a/stub/hnejpjg) | [Office 365+: Returns the relative position of an item in an array or range of cells.](https://support.microsoft.com/en-us/office/xmatch-function-d966da31-7a6b-4a13-a1c6-5a33ed6a0312) |

## Formulas for solving interesting problems

### Formulas that deal with doing time tricks

Is there a way to essentially make a counter so every 50 minutes it adds +1 to a total?

The NOW() function returns the current date and time. Dates in Excel treat full days as +1; for example today 26 December 2021 = 44556 and tomorrow's date will be 44557.

That and some division is enough to get this done. EG, =ROUNDDOWN((NOW()-44556)\*24\*60/50,0) counts the number of 50 minute intervals that have completed since 12/26/2021 12:00 AM.