13 Editing – information

Under this heading, I've grouped together macros that provide useful information about the bit of text you are working on.

Identifying the next character

(video: https://youtu.be/LAoxTjckzEE)

(Thanks to Marcela Robaina, there is now a Spanish-language version of this: *Chirimbolos*!)

Can you tell what each of these characters is: l|I1°º? Difficult, isn't it?

The differences are more obvious if I increase the font size: l|I1°º but in Century Gothic it's hard: l|I1°º

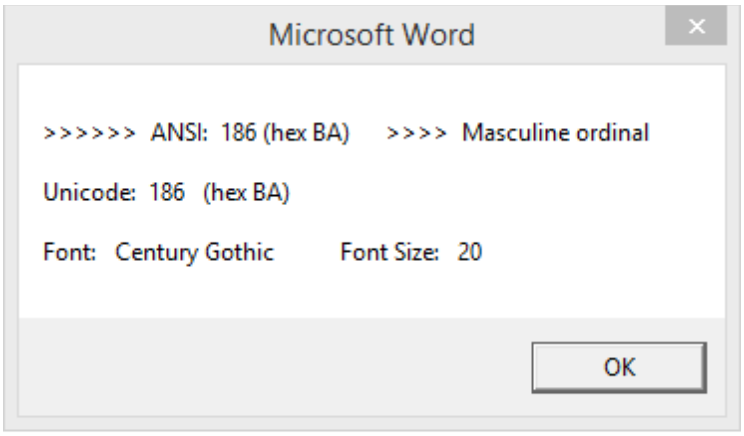
Similarly, can you tell the difference between − and –? If you put them between angle brackets you get a clue:

>−< and >–<

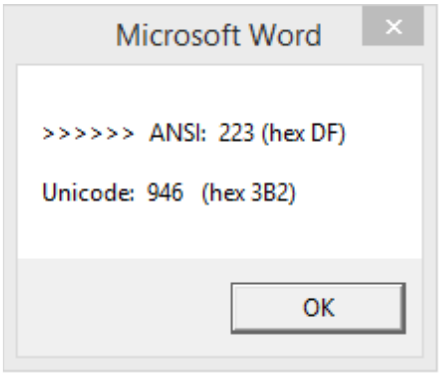
The first one is a proper minus sign, and the second is an en dash. The maths symbols are designed so as to line up horizontally: >−+=<.

The *WhatChar* macro looks at the character to the right of the cursor and tells you what each character is. So it will tell you for each of l|I1°º that they are a *lowercase l (el)*, a *vertical bar (vertical bar)*, an *uppercase I (eye)* and a *number one*, then a *proper degree symbol*, a *masculine ordinal* (as used in Nº6) and a *superscripted 'o'*.

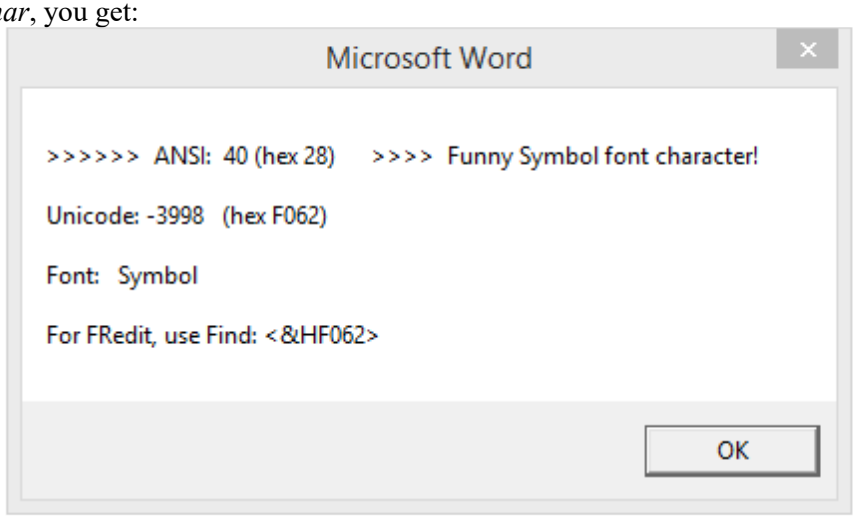
And while I was at it, I decided that it might as well give us more information about the character, so it also tells you if the character is super- or subscripted, and also what font it's in (but only if it's in a font other than the font used by Normal style.



What's more, it gives the Unicode number in hexadecimal as well as in decimal. So what?! Well, for example, a Unicode Greek beta (β) is displayed as 'Unicode: 946 (Hex 3B2)'. This is useful because if you type '3B2' (or '3b2') followed by an Alt-X, it turns into a beta character. (But watch out, because the 'B' can look a bit like an '8', depending on the screen font used, and its size on screen.)



If you get some of the old Symbol font characters and want to replace them with proper Unicode characters, I know of no way to find and replace them using Word's F&R. However, you can do so with *FRedit*. Here's an example – β – and if you use *WhatChar*, you get:



If you now click <ctrl-V> you get:

<&HF062>|

which is the start of a *FRedit* item. Just add a proper Unicode β:

<&HF062>|β

and when you run *FRedit*, all those nasty Symbol fonts betas will be changed to Unicode.

(The FRedit library, which comes with *FRedit*, has several of these set up for you already.)

Accented characters

In a document using unicode, a character such as a u-umlaut 'ü' can be produced by either be a single unicode (U00E1) or a pair of characters: an ordinary 'u' followed by a no-space umlaut accent, which then is displayed on top of the 'u'.

So WhatChar checks to see if the character ***after*** the character you're trying to check is a no-space character, in which case it beeps and warns you.

Now with added voice!

I've added the possibility of using voice so that it speaks the character, and then you don't need to clear the on-screen prompt window, giving you all the gory details; I find this speeds up the process . However, if you want those details, just select the character and run the macro again.

So to enable voice, you need to change to:

```
useVoice = True
```

and you also have to 'uncomment' the line at the beginning of the macro:

```
' If useVoice = True Then Set speech = New SpVoice
```

i.e. delete the apostrophe.

If you try this and Word complains: “Compile error: User-define type not defined” then you need to enable voice on your copy of Word (available from Word 2010 onwards, I think):

In VBA, click on Tools–References and find “Microsoft Speech Object Library”, tick the box and click OK. On my computer there are two lines saying “Microsoft Speech Object Library”, so make sure you tick the one that says ‘sapi.dll’ at the end, and not the one saying ‘sapi_one’.

Sub WhatChar()

Sub Chirimbolos()

Show any ‘funny’ codes in the text

(Video: youtu.be/_fWD4sXNg5s)
[investigate, strange characters, foreign characters]

The idea here is that if you suspect that the text has ‘funny’ codes in it, just place the cursor ahead of where you think the funny characters might be, and run the macro. It looks through the words, one at a time, and if it finds anything ‘funny’, it stops and indicates what it has found.

Now, ‘funny’ is initially defined as any ASCII code less than 32, or greater than 255 (i.e. Unicode). However, that includes things like end of paragraph (13) and tab (9), so you can decide whether to show those sorts of things by appropriate True/False settings in the first few lines.

You can also decide whether or not to show the Unicode numbers or not. And, since dashes are usually displayed as Unicode numbers, there’s an option *not* to show dashes even if you *are* showing Unicode numbers.

If you make the first line of the macro:

```
showEverything = True
```

then it will show everything. Then if you want to be selective, you can change this back to `False` and select the True/False appropriately for each feature, as mentioned above.

(Anecdote: The genesis of this macro was when I had a problem where the right-hand end of each line containing a note marker that was raised by about 3 pt, and no one seemed able to solve it. This macro revealed the following: in the place where it said, say ‘blah, blah.[6] Wibble wibble...’ what was actually there was ‘blah.[21][21][2][21][21] Wibble’, i.e. there was the expected ASCII [2] for the note marker (highlighted above), but it had some [21]s either side of it. These are apparently ‘closing field braces’. This was apparently some sort of debris remaining after the file had been multiply edited by various contributors. With a global F&R, I found ^21 and replaced it by nothing, and this solved the problem.)

Sub TextProbe()

Find next group of special (unicode) characters

This macro was written originally for locating Chinese characters within a text, but it could be used for Arabic or other language characters, indeed for any ‘special’ characters.

It’s just a ‘Find this’ type macro which uses wildcards to find any character(s), by specifying ‘characters within this range’, just as [a-z] would find any lowercase character.

The characters can be selected by their hexadecimal unicode numbers. What? Eh?! Well, if you use the FRedit list below, it will highlight, in various colours, the different ranges of characters that you might be interested in:

```
~[<&H1000>-<&H1FFF>]|^&
~[<&H2000>-<&H2013>]|^&
~[<&H2023>-<&H2FFF>]|^&
~[<&H3000>-<&H3FFF>]|^&
~[<&H4000>-<&H4FFF>]|^&
~[<&H5000>-<&H5FFF>]|^&
~[<&H6000>-<&H6FFF>]|^&
~[<&H7000>-<&H7FFF>]|^&
~[<&H8000>-<&HFFFF>]|^&
```

And here's a piece of text, which was highlighted by (the originalk version of) the above:

白凱琳 / コリーン・ベリー, 博士

The bullet in the middle (yellow) is a unicode in the range 2000–2FFF, that you probably don't want to find, so I split the range into two, so as to avoid common symbols such as open and close single and double quotes and em and en dashes.

But the Chinese-type characters are in the ranges 3000+, 5000+ and 7000+, so in using the macro, I'd probably tell it to look for 3000–7FFF. So at the beginning of the macro I'd put:

```
myRange = "3000-7FFF"
```

(Interestingly, the 'comma-and-space' is actually a *single* unicode character, FF0C (as revealed if you use my WhatChar macro.)

Ha! Having said to use 3000-7FFF, the editor came back to say the macro had missed some characters. When I checked with WhatChar, they were in the 8000+ range, so I got her to change to 3000–8FFF.

And now I've improved the macro so that it ignores common characters we use, unless they happen to occur within a group of more obviously 'special' characters.

```
Sub FindSpecialCharacters()
```

Spellcheck a single word

(Video: youtu.be/W-JX3P1hZF8)
This macro speeds up the process of spellchecking a single word. There's no need to select the word; just place the cursor anywhere in the word and press whatever key combination you have assigned to the macro. If the word is correctly spelt, the computer beeps, but if it is incorrect, it opens the spelling dialogue box for you to correct the spelling.

If the beep annoys you, remove the line that says `beep`; you will know that the spelling check has been run because the cursor jumps to the start of the word.

Another small timesaver is that, unlike using F7, after it has done the spellchecking and found that the work is OK, it does not then ask if you want to spellcheck the rest of the document.

The macro comes in three 'flavours', one that spellchecks in UK English, one in US English, and the third in whatever the current language of the document is.

And now a fourth: spellcheck the word in the other language, i.e. if the current language is UK, check it in US, and vice versa.

```
Sub SpellcheckWordUK()
```

```
Sub SpellcheckWordUS()
```

```
Sub SpellcheckWordCurrent()
```

```
Sub SpellcheckWordUSUK()
```

Spellcheck with language warning

How many times have I started a spellcheck and then, in the middle somewhere, it throws up, say, 'flavour' as a spelling error. Drat! I hadn't noticed that the language set for the document is US English.

This macro checks the current language and, if it's UK English, it just carries on with the spellcheck as normal, but if it's US English, it beeps at me first, and then does the spellcheck.

So, all you do is to allocate this macro to the F7 key, so that everything work as as normal, apart from the warning beep for US language files.

```
Sub Spellcheck()
```

This next version is, perhaps more useful. What happens is that if it's *all* in US English, it'll just beep at you, but if there's a mix of languages, it will throw up a warning message.

```
Sub SpellcheckWarn()
```

Spellcheck and auto change

This macro checks the spelling of the word at the cursor. If it's OK, it beeps and moves on. If it's not a correct spelling, it replaces the word with the first alternative that Word offers. However, if there are no alternative spellings offered, it highlights the word instead. (Remember that, for either change, you can just use Ctrl-Z to undo it.)

```
Sub SpellWordChange()
```

(This macro gets a mention in video: youtu.be/FVt2ggFXf4A)

Another similar macro is *SpellingSuggest* which simply changes the spelling of the word at the cursor to Word's suggested replacement. This macro doubles for use with *FRedit* – see above in the Pre-editing Tools section.

```
Sub SpellingSuggest()
```

Delete all spelling errors in a file

(Video: youtu.be/AqREu_iJ2Yg)

I can't remember why, but someone wanted to delete all the incorrectly spelt words from a file. That's what this macro does!

(Now updated, on request from the user, so that if an area of text is selected, it only works on that part of the text, not the whole of the document.)

```
Sub DeleteAllSpellingErrors()
```

Show (or not) spelling errors in a file

You may, like me, not like having all those red wiggly lines under words in your text. In which case, it's nice to be able to switch them on and off quickly and easily. That's just what this macro does.

```
Sub SpellingShowToggle()
```

Count this word/phrase

(video: https://youtu.be/LAoxTjckzEE or https://youtu.be/guPVq57Vgm4)

This macro allows you to select some text, and it will tell you how many times that word/phrase occurs in the whole text (now including the footnotes and endnotes).

Not only does it do a straight count, but it also does a case-sensitive count, a count for italic/bold/bold-italic versions thereof and even a whole-word count.

So, for this document, if you select ‘et al’, it gives you:

All: 24
Case sensitive: 24
Italic: 4
Bold: 2
Wholewords (case sensitive): 22

i.e. it finds things like ‘my pet alligator’, and knows they are not proper ‘et al’s.

If you don’t select a word or phrase, it will assume that you want to search on the word at the cursor.

The ‘phrase’ can even include newlines, so you could, for example, count the number of times a word occurs at the start of a line.

If you don’t want the facility to count the different bold/italic variations, they can be switched off in the first line of the macro, and if you don’t want it to spend time doing a whole-word count (which on a long document with foot/endnotes can take some time), that can be switched off in the second line of the macro.

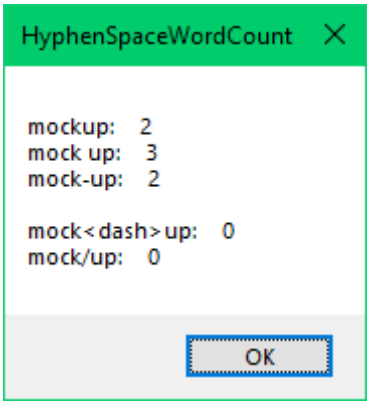
However, in this latest version, I’ve added a feature whereby you can set the maximum time that you’re willing to wait for an answer (`maxTime = 0.5` means half a second), so if the macro has time within that limit, it will do more and more of the different counts. If it ran out of time, it will tell you so.

```
Sub CountPhrase()
```

Count hyphen/space/single word

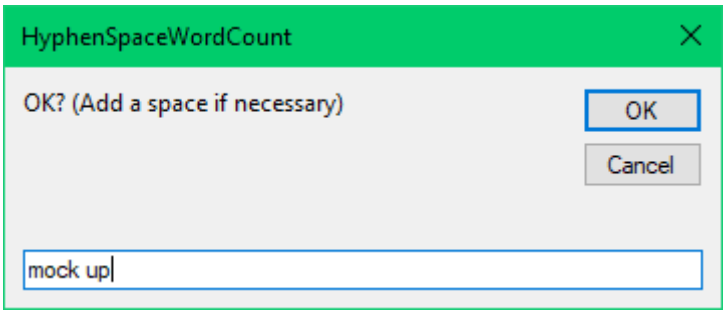
(Video: https://youtu.be/hqPVJSZsFDk, Later video: https://youtu.be/LAoxTjckzEE)
(Even better video: “Super-Searching 4” (7:49) https://youtu.be/m4gVuqrl83w)
(Or https://youtu.be/guPVq57Vgm4)

The idea here is that if you are reading the text and you see a word such as ‘mock-up’ (which could equally appear as ‘mock up’ or ‘mockup’) and you want to find out what the author has used predominantly, you just click in ‘mock’ and then shift-click in ‘up’, and run this macro. It will count the numbers of each variant.



(As you can see, it also checks for ‘mock–up’ and ‘mock/up’, at no extra charge!)

What if it’s ‘mockup’ that’s at the cursor? Click in the word (don’t select it) and run the macro; it does its best to guess where the split should come?



Change it if necessary and press Enter.

```
Sub HyphenSpaceWordCount()
```

Count words remaining

(video: https://youtu.be/LAoxTjckzEE)

This macro counts the number of words from the current cursor position, down to the end of the document, to give you a feel of how much more reading there is to do!

There’s a simple version:

```
Sub CountRemainderSimple()
' Version 25.04.16
' Count words below the cursor

wordsTotal = ActiveDocument.Content.Words.Count
Selection.End = ActiveDocument.Content.End
wordsLeft = Selection.range.Words.Count
Selection.Collapse wdCollapseStart
perCent = Int(1000 * (wordsLeft / wordsTotal)) / 10

MsgBox (perCent & "% left.   (Very roughly " & _
     Int(0.0007 * wordsLeft) & " thousand words)")
End Sub
```

Unfortunately, the method used to do the count (`Words.Count`) treats even items of punctuation as ‘words’, so it only gives a very rough indication.

MS Word also has some statistics functions which give a much more accurate count of the number of words (the same as displayed on the status bar at the bottom of the window). However, these statistics refuse to work if the file has bits of text that have been set to different languages.

So, when using this more complicated macro, if it finds multiple languages, it asks you if you want to standardise to either UK or US English. If you need to preserve the multiple language settings, then you can't use this macro, but if you say 'Yes' when it asks if it's OK to set the language, then it gives an accurate word count.

I like CountRemainder to display as '23.5', rather than '23,521' because I 'think' in thousands of words, but if you prefer the latter, alternative display format then there's now an option at the beginning of the macro:

```
' altDisplay = True
altDisplay = False

Sub CountRemainderSimple()

Sub CountRemainder()
```

Count italic text

One reader has a job involving an nth edition update of a book where they put all the new text in italic. Payment is on the basis of one rate for repeated text and a higher rate for new text. So the idea is to count the number of characters in italic and then the rest are roman.

However, you need to take account of the fact that there may be text in footnotes, endnotes and textboxes. These are also counted.

```
Sub ItalicCount()
```

Count words within sections

If you want to know how many words there are in each section of a document, you can use the style of the section heading – say, Heading 1 and this macro will count the words between one section heading and the next.

It counts them all and then creates a new document, listing the section titles and the number of words in each. For example (I've scraped a bit of the text around here, and put it in a separate file, to generate...):

Before first heading	33
Count words that are highlighted	56
Count words within sections	69
Copy paragraphs that contain highlighted (and coloured) text	55
Highlighting words not in vocabulary list	108

The first line is the count of the words at the head of the document, before the very first Heading 1 item in the document. If it's a big document of a number of chapters then you might change the first line of the macro from:

```
myStyle1 = "Heading 1"
```

to, say:

```
myStyle1 = "Chapter title"
```

If you want to count the text between both of two headings, then use, say:

```
myStyle1 = "Heading 1"
myStyle2 = "Heading 2"
```

And if you use two style, and want the first style to be in bold in the list, use:

```
    doBold = True
```

```
Sub CountSectionWords()
```

Count pages within chapters

If you want to know how many pages there are in each chapter of a document, you can use the style of the chapter heading – say, Heading 1 – and this macro will count them.

```
Sub CountChapterPages()
```

List all the italic words

(Video: youtu.be/AqREu_iJ2Yg)

This macro first uses the macro, *CopyTextSimple* (assuming you have it loaded in your computer), to create a new file with a copy of all the words in the file. Then it simply deletes all the non-italic text, so you're left with a list of all the italic words and phrases.

I wrote it because an author wanted me to make suggestions for words to put in his glossary. Fortunately, he had used italic to highlight all new words as they appeared, so it was very easy. Once the list was created, I used SortIt to sort it into alphabetic order, then DuplicatesRemove to leave a unique list of words and phrases. (Oh, I could have used SortAndRemoveDups, which does the two jobs in one!)

```
Sub ItalicWordList()
```

Multifile word counting

This macro looks at the files within a particular folder and loads all the .doc, .docx and .rtf files, one by one, and counts the total number of words.

As with my other multifile macros, to get you to identify the folder containing the files, the macro brings up the Open File window, so you navigate to the required folder and then click Cancel. The macro then generates a list of all the files in the folder and asks whether you want to work on all the files in the list. If you say 'Yes', it uses the complete list of files that it has created and works through them all one by one.

If you say you *don't* want to work on all the files, the macro just stops. The list will look then something like this:

C:\Program Files\VirtualAcorn\VirtualRPC-SA\HardDisc4\MyFiles2\WIP
Macro Jobs.doc
Roman cats.doc
Stats.docx

The point is that you can then edit this list, either deleting files you don't want included, or putting a vertical bar ('|') in front of any you want it to ignore.

If you now run the macro again, it recognises that this Word document is a file list and so it proceeds to work through the listed (and not ignored) files, opening each one and counting the contents.

Now we need to think about the actual counting – it's not straightforward!

As you may be aware, Word's word counting is – shall we say – 'idiosyncratic', in that different ways of counting give you different answers. This macro therefore (a) does the count using the 'readability statistics' and (b) conditions the file by getting rid of all the punctuation in an appropriate way and then counting the number of words with the `Words.Count` command. For example, 'can't' and 'whole-hearted' are treated as single words, whereas 'either/or' is treated as two words.

The macro also copies the text out of all the textboxes, plus the text of the foot/endnotes, and counts that too (by each of the two methods). This it refers to as ‘extra’ text.

It then presents you with a complete listing of all the files, showing the ‘number of words’ according to the two methods – which it refers to as ‘(stats)’ for ‘stats’ and ‘(count)’ for counted – for the main text, the ‘extra’ text and the whole of the text in each file.

I’ve tested a range of files, looking at the ‘stats’ value, the ‘counted’ figure and the values given by <Alt-W>. The conclusion I’ve come to is that it’s impossible to say which is the ‘correct’ value (or least inaccurate). Apart from anything else, is ‘1996’ a ‘word’? Is ‘1996-98’ one word or two? Is ‘i.e.’ a word? Try this: open a new document, type ‘i.e.’ a few times, or ‘1996’ and confirm that each is indeed treated as one word. Now, copy this current paragraph and paste it into your new document. Note the word count and then try deleting either ‘i.e.’ or ‘1996’. When I tried it, *sometimes* the word count dropped by one, but at other times it remained unchanged.

Also, what about section numbers: ‘1.3.4 Title of this subsection’, or whatever?

At the beginning of the macro, you can select to use either stats, or count or both.

```
useStats = True
useBoth = True
```

If the files contain equations – either MathType or Equation Editor – then it counts those too. You can disable this feature by using:

```
countEquations = False
```

Sub MultiFileCount()

Totalling words from various places

Someone asked on SfEPLine about adding up the number of words in quotes. Just for fun, I knocked up a macro that allows you to marks bits of text and it will add them to a list, totalling them as it goes, so you end up with something like:

```
 89     p.5 – So wrote Hugh
 119    p.14 – It sounds like
 18     p.176 – Ordering is utterly
 114    p.176 – Similarly: in
 87     p.177 – Author: You
 29     p.178 – Macrostate indifference is
 51     p.179 – Dominance lemma:
 82     p.97 - 89
=====
 507
```

It doesn’t total the numbers; it just adds in the next item that you’ve selected, giving the number of words selected, the page number and the first three (or whatever number you choose) words of the selection.

If you add something in error, you can simply Ctrl-Z it off the list to take you back to where you were before you added it.

There’s no need for you to create a totals file as it adds one if one doesn’t exist already.

Sub WordTotaller()

Count all the words by Heading 1 and Heading 2

This counts all the words in each section of a document, based on headings 1 and 2, and it comes out like a contents list, but with word numbers, not page numbers. The counts include the footnotes and/or endnotes that are cited within that section:

Prelims 10
The Evolution of Modern Philosophy 1241
Historical Foundations 434
Contemporary Developments 339
Future Perspectives 463
Scientific Methods and Applications 759
Experimental Design 272
Data Analysis 262
Research Implementation 221

Total word count 2010

```
Sub WordCountByHeading()
```

Check the column totals

I just needed to check lots of figures in tables such as:

Number	Per cent
4	1.7
45	18.7
103	42.7
72	29.9
13	5.4
4	1.7
241	**100**

Now in each table, the figure at the bottom of each column should be equal to the total of the figures above. So if you put the cursor in the first cell of the column, it adds up the figures until it drops off the bottom and then checks back to see that the final figure was indeed the total of the other.

Because there are likely to be rounding errors, I have allowed the user to set an accuracy:

```
allowErrorPercent = 0.01
```

so if the error is less than 0.01%, the macro just beeps to tell you it's OK, but if the error is more than that it tells you what it thinks the total should be, and you can act accordingly – raise an author query or whatever.

The `okChars` line allows some of the cells to contain things like en and em dashes, and also full points (periods) without thinking that it has reached the end of the column.

If a box has a zero value, then as long as it has a hyphen or dash or an actual zero then it carries on looking for the end of the column of figures. However, if a cell is completely blank, it assumes that it has reached the end of the column.

```
Sub ColumnTotal()
```

Check the totals of a set of consecutive numbers

If you select any range of text that includes various numbers, even if the numbers include commas separating the thousandss and/or decimal points, the macro will total all of the numbers for you. However, it will also check to see

whether the first number is the sum of the remaining numbers or if the final number is the sum of the others, in which case it beeps to reassure you that all is well.

If it can't find a summation figure, it simply tells you what the sum is.

```
Sub NumberTotaller()
```

What is the full filename?

For some macros, you need to put into the macro the address of some file that it uses. To make this easier, open the Word file in question, and run this macro. It copies the full filename, so you can then go back to the macro and paste the filename in wherever it is needed.

```
Sub FullFileNameCopy()
```

Get information from Google etc

(Video: youtu.be/RYggCNcK-h8, Later video: https://youtu.be/LAoxTjckzEE)
(GoogleMapFetch: https://youtu.be/MgW7x_BOG3c)

(See the following section for how to set up your own macro for accessing your favourite website.)

N.B. With most of these macros, if you select nothing, it will launch the current word; if you do a rough selection of a range of words, it will try to round off the selection to include the first and last words in the range.

I discovered that it is possible to launch a URL from within a macro, so I realised that if I selected a word or phrase, I could look it up straight away on Google. What's more, I could have a second version of the macro that puts quotes around the selected phrase before sending it off to be Googled. Then I thought – yes, and with Wikipedia. Then I thought – yes, and with OUP's online dictionary, etc, etc.

N.B. If Google throws up a prompt about accepting all cookies, the macro will fail to paste in the traget word/phrase. In which case, you'll have to use **GoogleFetchCookie**, which jumps to the 'AccepAll' button and introduces a delay to allow the pasting to take place.

For *GoogleFetch*: Click in a word (or select some text), but there's no need to select exact whole words; the macro will expand to the nearest word end, e.g.

London marathon

will send "London marathon" to Google.

Ditto for *GoogleFetchQuotes*, etc.

Currently GoogleFetch is set to Google UK. Simply replace the text **in between** the quotation marks in the fourth line:

```
mySite = "https://www.google.co.uk/search?q="
```

Here are some country-specific URLs:

Australia: `https://www.google.com.au/search?q=`
Canada: `https://www.google.ca/search?q=`
India: `https://www.google.co.in/search?q=`
Ireland: `https://www.google.ie/search?q=`
New Zealand: `https://www.google.co.nz/search?q=`
South Africa: `https://www.google.co.za/search?q=`
US: `https://www.google.com/search?q=`

For Google Scholar: `https://scholar.google.com/scholar?q=`

There are also now some German versions (and now Dutch ones below).

For a speedy search of **GoogleMaps**, just click in a word (or roughly select some text – the macro rounds off the selection), run *GoogleMapFetch*, and the word or text will be launched to GM.

And if you want to go from home to Hull, type "h to Hull" or just "h/Hull" and run the macro. Or to go from another frequently used address (say your work) to Timbuktu, type "w/Timbuktu" and run the macro. Or for place-to-place searches use "harwich to hull", or "harwich/hull".

You could even set up a round trip, say: `h/leeds/s6 6ru/oxford/h`.

The postcodes for 'h' and 'w' are set at the beginning of the macro.

```
myHome = "NR8 6TR"
myWork = "M21 0UW"
```

And I guess it would work with Zip codes or whatever postal code you use in your country.

You might possibly want to change the site address to your own country, currently:

```
mySite = "https://www.google.com/maps/dir/"
```

so then, for Egypt, you'd change it to:

```
mySite = "https://www.google.com.eg/maps/dir/"
```

For NgramFetch: Click in a word (or select some text), run *NgramFetch*, and the word/text will be launched to Ngram. If you want to compare two (or more) words/phrases, separate them with commas:

compared to, compared with

There's no need to select exact whole words; the macro will expand to the nearest word end:

compared to, compared with

The Year start and Year end of the display is set at the beginning of the macro:

```
yearStart = "1800"
' yearStart = ""

yearEnd = "2000"
' yearEnd = ""
```

If you want to search up to the current year, use:

```
yearEnd = ""
```

Also added, a fetch macro for the Maori dictionary, *Te Aka*.

Sub GoogleFetch()

Sub GoogleFetchQuotes()

GoogleFetchCookie()

Sub GoogleBooksFetch

```
Sub GoogleScholarFetch()

Sub GoogleFetchDE()

Sub GoogleFetchQuotesDE()

Sub GoogleFetchUS()

Sub GoogleFetchQuotesUS()

Sub GoogleFetchCA()

Sub GoogleFetchQuotesCA()

Sub TeAkaFetch()

Sub WorldCatFetch()

Sub CollinsFetch()


Sub NgramFetch()

Sub OUPFetchPremium()

Sub OUPFetchPremiumMac()

Sub OxfordPremiumThesaurusFetch()
```

Google Maps

```
Sub GoogleMapFetch()
```

for *wikipedia.com*

```
Sub WikiFetch()
```

for *thesaurus.com*

```
Sub ThesaurusFetch()
```

for *openthesaurus.de*

```
Sub ThesaurusFetchDE()
```

for *dictionary.com*

```
Sub DictionaryFetch()
```

for *duden.de*

```
Sub DictionaryFetchDE()
```

for the Australian Macquarie dictionary

```
Sub MacquarieFetch()
```

And if you use the Premium Macquarie, you'll need to login first. Try this macro. However, you have to put your username and password into the macro. Also, the delay that the macro uses – which allows the website to respond, before typing in the username and password – might need to be increased to mayb 1.5s or even 2s.

```
Sub MacquariePremiumLogin()
```

Spanish medical dictionary
Sub DTMEfetch()

Spanish language dictionary
Sub RAEfetch()

Mormon (Latter-day Saints) site:
Sub LatterDayFetch()

Bible verses
Sub BibleGatewayFetch()

Sub BibleHubFetch()

The former has a line:

```
myVersion = "NIV"
```

which you could change to NKJV or ESV etc. Or use `myVersion = ""` if you don't want to specify.

With these two macros, you can carefully select the reference, but if you just place the cursor in the (abbreviation of the) name of the book they do their best to select the numbers and punctuation before and after the name, e.g. with "1 Cor 13:8", click in "Cor" and it should pick it up OK. If you find that with the particular punctuation you try it with it doesn't work, please let me know, and I'll try to fix it.

Sadly, I can't get the latter macro to work fully. After the webpage comes up, you will need to click Ctrl-V and press Enter to complete the fetch. However, it does have the advantage that you can select, say "greatest of these is love", and it will find 1 Cor 13:13, whereas the Gateway doesn't offer that possibility. (Mind you, GoogleFetch tells us that the quote is from 1 Cor 13:13, anyway!)

(And, based on an idea from Rob Worth...)

If your document has different texts in different languages, then as long as those bits of text are set to the relevant language, then this macro sends the current word off to different dictuionaries, depending on the language set.

Which dictionary is used for each given language is set within the macro:

```
Select Case myLanguage
  Case wdEnglishUS
       mySite = "https://www.merriam-webster.com/dictionary/"
  Case wdEnglishAUS
       mySite = "https://www.macquariedictionary.com.au/features/word/search/"
  Case wdGerman
       mySite = "https://www.duden.de/suchen/dudenonline/"
  Case wdFrench
       mySite = "https://www.collinsdictionary.com/dictionary/french-english/"
  Case Else
       mySite = "https://www.lexico.com/definition/"
End Select
```

If you want to add extra languages, you will need to follow this same pattern. If you can't work out what to do for any given language, just get in touch.

Sub DictionaryFetchByLanguage()

Sub ReversoFetchMultilingual()

for *PubMed*
Sub PubMedFetch()

for *OneLook*
Sub OneLookFetch()

for *Merriam Webster*
Sub MerriamFetch()

And for *Merriam-Webster legal*, you can use:

Sub MerriamLegalFetch()

And if you have **subscriber access** to **Merriam-Webster unabridged**, you can use:

Sub MerriamCollegiateFetch()

Sub MerriamUnabridgedFetch()

Sub MerriamMedicalFetch()

Sub MerriamThesaurusFetch()

For the *Dictionary.Law* website:

Sub LawDictionaryFetch()

British Library catalogue website:

Sub BLcatalogueFetch()

When my son moved to Paris, I tried to get my O-level French back from the dead, so I've started using Google Translate. *(Quand mon fils a déménagé à Paris, j'ai essayé d'obtenir mon français O-level de la mort, donc j'ai commencé à utiliser Google Translate.)*

This macro looks at either the currently selected text, or, if there's nothing selected, the current paragraph, and delivers it to Google Translate. However, before it does so, it spellchecks the first word of the text and, if it's a spelling error in both UK and US English, it assumes that it's in French, and translates it to English.

Google Translate offers French and Spanish (and many other languages), so if you want the latter, at the beginning of the macro use:

```
myLanguage = "es"
```

Sub GoogleTranslate()

Some Dutch colleagues asked for Netherlands-related versions, so here you go:

Sub WikiFetchNL()

Sub GoogleFetchNL()

Sub GoogleFetchQuotesNL()

Sub ThesaurusFetchNL()

Sub DictionaryFetchNL()

Sub GroeneBoekjeFetch()

For Chicago Manual of Style, if the first macro below works for you, use that; if not, try the following one.

(Mac users! Sadly, your computer may not have the special VBA command that this macro requires – drat! So you will have to run the macro, which will select and copy your search item, and then open the web page, but you will then have to click in the search box yourself, and click Cmd-V.)

Sub CMOSFetch()

Sub CMOSFetchAlt()

Set up your own Fetch macro (1)

(Only use this macro if there isn't a macro above, already set up for you.)

Open your browser and go to your chosen website; now type a suitable word into the search box and click Enter.

If you look at the browser's URL, you'll see how the it was formed from your chosen word(s), for example:

`https://www.dictionary.com/browse/elephant`

or

`https://www.google.co.uk/search?q=elephant`

The principle now is that you give the macro the bit to the left of your chosen word, and then the macro, when you run it, will combine together (a) the URL of the site with its search function (e.g. '`search?q=`', if it needs it) and (b) the text you happen to have selected.

So, in VBA, you have to create a copy of this generic *SomethingFetch* macro, giving it a sensible new name, and then copy 'the green bit' above and paste it into the first line of the macro, to give something like the following (but note that, for my two example websites, there is already a macro ready to use on my website):

```
Sub DictoFetch()
' Paul Beverley - Version 27.08.22
' Launches selected text to dictionary.com

mySite = "https://dictionary.com/browse/"
```

or

```
Sub GoogleFetchUK()
' Paul Beverley - Version 27.08.22
' Launches selected text to google.co.uk

mySite = "https://www.google.co.uk/search?q="
```

That's the theory, and it creates the most reliable Fetch macros. However, for a variety of reasons, it doesn't always work, especially for websites that have password-protected access. If you can't get it to work, try method 2.

Sub SomethingFetch()

Set up your own Fetch macro (2)

(Only use this macro if you've tried macro (1) above, and you can't get that to work.)

(Sadly, it seems as if this option is **not open to Mac users** as Word for Mac doesn't support the `SendKeys` command that it uses, sorry.)

This method requires a lot of fiddling to get it to work, though once you've nailed it, of course, you've got a macro that will save you lots of time.

The principle is this: When you access a search site manually, you open the page, click in the search box, and paste in the word(s) you've copied from your Word file, and click Search (or press Enter).

The macro's job is to simulate your manual actions, but it can't simulate mouse clicks, so you have to do that search again, but this time use the Tab key to get into the Search box.

~~So when you try this~~, count the number of times you have to click it – mind you, on some (friendly) sites you find that the cursor is already in the Search box, and you don't need to press Tab.

When you know how many Tab presses you need, create a copy of this dummy macro, giving it a new name, and enter the address of the web page, and also the number of times you had to click the Tab key:

```
Sub MySiteFetch()
' Paul Beverley - Version 27.08.22
' Launches selected text to mysite.com

mySite = "https://dictionary.com/browse/"
numTabs = 5
```

Then when you run the macro, it reads the text from Word file, copies it and then launches the website according to the URL you give it, and passes the Tab characters through to the keyboard, plus a Ctrl-V (for paste) and then an Enter – all as if you had typed them in yourself.

The trouble is, this often doesn't work either! What the macro has to do is 'count to ten' before sending off the key codes. The reason is that if you pass the characters to the keyboard too soon, the web page is not ready for them and it misses them, so nothing happens!

So how long do you have to wait? Well, it just depends, so you have to try it out and see. Here are a few examples from having tried various different webpages. (N.B. The sites I have used as examples all have their own dedicated macros anyway, so rather use those. Only use this macro if you really can't get the version (1) to work.)

```
Sub SomethingFetchAlt()
' Paul Beverley - Version 27.08.22
' Launches selected text to Xyz website.

' mySite = "https://www.merriam-webster.com/": numTabs = 0: myWait = 0.9
' mySite = "https://google.com/": numTabs = 0: myWait = 0.5
' mySite = "https://dictionary.com/": numTabs = 11: myWait = 2
' mySite = "https://thesaurus.com/browse/": numTabs = 11: myWait = 1.5
mySite = "https://www.macquariedictionary.com.au": numTabs = 0: myWait = 2.5
```

So `numTabs = 11` means 'type 11 Tab characters', and `myWait = 2` means wait for 2 seconds before typing in the tab characters. As you see, the longest wait I had to give was 2.5 seconds, and three of the sites I tried didn't need *any* tab characters.

So when you try it, start with, say, 4.5 seconds to be sure it will catch the characters you send to the keyboard, and then try reducing it, as you want as short a delay as possible, for speed.

Sorry it's so complicated. If you try a site and can't make the macro work, do get in touch.

Oh, and bad news for Mac users! Sadly, your computer may not have the special VBA command that this macro requires – drat! So you will have to run the macro, which will select and copy your search item, and then open the web page, but you will then have to click in the search box yourself, and click Cmd-V.)

```
Sub SomethingFetchAlt()
```

GoogleFetch using a specific browser

As it stands, GoogleFetch just says 'look this up on a browser', and the computer uses the default browser, as set up at some stage by you. However, this macro to sends your Google URL with your chosen text to a ***specific browser***, and not just the default browser.

I wrote this macro because an editor said he liked to have one browser with various tabs running for access to specific resources, but when looking up things on Google, he wanted to use a *different* browser. Thus his main browser wasn't 'clogged up' with lots of Google fetches.

The browser that the macro uses is set up at the beginning of the macro. I've su8ggested what I think the likely browser address is on your computer, but you might need to check and adjust accordingly for the browser you want to use:

```
runBrowser = "C:\Program Files\Mozilla Firefox\Firefox"
' runBrowser = "C:\Program Files\Google\Chrome\Application\Chrome"
' runBrowser = "C:\Program Files\Internet Explorer\iexplore"
```

```
Sub GoogleFetchSpecificBrowser()
```

Fetch from multiple dictionaries

Select a word (or just click in it) and run this macro to look up the word on three different dictionary websites: it is set for dictionary.com, oxforddictionaries.com (Lexico) and collinsdictionary.com, but you could change the options at the beginning of the macro if you have another go-to site that you want to use.

```
Sub DictionaryMultipleFetch()
```

Using Google to search a specific site

Google has a command that allows you to search one specific site, so if you want to find out, say, about the courses on creative writing that UEA runs, you can type into Google:

creative writing site:uea.ac.uk

So if you have a site you often want to search, you can put it into this macro.

At the beginning of the macro is:

```
' mySite = "wordmacrotools.com"
' mySite = "archivepub.co.uk"
mySite = "uea.ac.uk"
```

As it stands, I would just type 'creative writing', select it, and run the macro.

So, if you use `mySite = "wordmacrotools.com"` then all our macros will be searchable.

```
Sub GoogleSSFetch()
```

Launch successive URLs from the text

This macro allows you to launch a URL from a Word file into your browser – just click to the left of the URL and run the macro. But if you select a bit of text just before the URL, e.g. double-click the word prior to the start of the URL, it will launch that URL, but also the next few URLs that occur in the text.

When you run the macro, it will ask you how many URLs to launch, but a default number will be offered. This is set at the beginning of the macro, so you can change it:

```
numberOfURLs = 10
```

Note that if the file contains hyperlinked URLs, because these may be invisible, the macro will first select the text from the cursor to the end of the file, create a new file, copy this selected text to the new file and then turn all the URL links into visible URL text.

So to save time, you can use this new file to launch the next set of, say, ten URLs.

The macro has the option to highlight all the URLs as it sends them to the browser, so you can tell which URLs you have already tested.

```
highlightURL = True
myHighlight = wdGray25
```

If the browser says there's an error in the URL, the macro will report this back to you. It will select the recalcitrant URL, so you can try copying it and pasting it manually into the browser; the browser will then tell you what error was generated – usually it's: Error 404: File Not Found.

However, look carefully at the selected text. While developing the macro, one URL was as shown below, and the selection was as per the grey highlight:

www.communityservices.nd.gov/uploads%5Cresources%5Cstrawbale.pdf

I therefore realised that the macro was failing to allow "%" to be a recognised character in the URL, so I added "%" into this line (the "%" is highlighted, so you can find it):

```
acceptableChars = "/a-zA-Z0-9.,;:=&#\?\(\)\[\]_+\-%" _
     & ChrW(8211) & ChrW(8212)
```

Now, if you discover a URL with a "funny" characters that the macro doesn't recognise, please add it into that line, after the "%". But also, please tell me what the character is, so that I can add it into the published version of the macro. Thanks!

```
Sub URLlauncher()
```

Turn URL or email address into a live link

Click anywhere in a URL or email address and run this macro. It selects (only) the URL or email address, and creates it as a hyperlink.

```
Sub URLlink()
```

URLs to active links

This macro finds all the URLs in the current file and makes them active links, i.e. clickable.

```
Sub URLlinker()
```

Reduce the text extent of an active link

If the text before was:

> This is explained in Post-Brexit Funding for Nations, Regions and Local Areas.

If you now select “Post-Brexit” and run the macro, it becomes:

> This is explained in Post-Brexit Funding for Nations, Regions and Local Areas.

Or if you had selected “Nations”, it would have given you:

> This is explained in Post-Brexit Funding for Nations, Regions and Local Areas.

Sub URLshrinker()

Unlink all the URLs

This macro unlinks all the URLs in the selection of the whole file. If the link is, for example, Free macros, where the URL is hidden, then it converts this to:

> ...for example, Free macros (https://www.archivepub.co.uk/macros.html) , where the URLis...

but the wording is no longer a clickable link.

But if the link is just the URL then just the URL is there as text.

There’s an option for making the text bold (which someone wanted), and you can remove the “https://” or “https://” part of the URL when displayed, viz. use:

```
remove1 = "https://"
remove2 = "https://"
```

Sub URLunlinker()

Email addresses to active links

This macro finds all the email addresses in the current file and makes them active links, i.e. clickable.

Sub EmailLinker()

Show all formatting or just paragraph marks

The heading says it all: I reckon it’s sometimes useful to be able to see the paragraph marks (especially if you suspect that there are some line breaks around), but putting **all** the formatting marks on is a bit too much – for me, it makes the text too difficult to read. This macro lets you toggle between showing all the formatting marks, just the paragraph marks and no marks at all.

If you want it to show both paragraph marks **and** tabs, change the first line to:

```
tabsToo = True
```

Sub ShowFormatting()

Show various formatting marks and hide highlighting

This is similar to the previous macro, but you can select, from a list, which format marks you want to show, and you also have the option of hiding the highlighting. All you have to be able to do is add up! You get this menu:

1 = show paragraphs
2 = show spaces
4 = show tabs
8 = don't show highlighting
16 = show hyphens
32 = show bookmarks

So, if you just want spaces and paragraph marks, it's 3 (2 + 1), and if you just want to show optional hyphens and tabs, it's 20. If you want to be able to see the text more clearly, you can choose ***not*** to display the highlights, by adding 8 to your number. If you want them all, to save you adding them all up, just enter 99.

To return the display to normal (no formatting marks, and highlighting showing) either type 0 (zero) or just press <Enter>.

I was getting frustrated that I (almost) always want option 7 (but sometimes 1), so I've also added a feature so that I can specify a couple of favourite options.

```
myFavouriteOption_1 = 7
myFavouriteKey_1 = "/"
myFavouriteOption_2 = 1
myFavouriteKey_2 = "#"
```

I happen to use <Alt-/> to run the macro, so I've chosen the slash key to give my normal option (7) and the hash key for 1. So it's just <Alt-/></><Enter> to switch on and <Alt-/><Enter> to switch off, or <Alt-/><#><Enter> when I only want to display the paragraph markers. You can, of course, choose different keys and different favourite options.

```
Sub ShowFormattingMenu()
```

Show all text in a different font size

(Video: youtu.be/P-6VdmT2BbE)

This is useful where you suspect that some parts of a section of text might be in a slightly different font size. It's quite difficult to tell sometimes, especially on things like punctuation.

Select the suspect section of text and the macro will assume that the correct size is that of the first word of the selected text. So, did you spot the rogue font sizes in the paragraph above? Here they are:

> This is useful where you suspect that some parts of a section of text might be in a slightly different font size. It's quite difficult to tell sometimes, especially on things like punctuation.

```
Sub HighlightNotThisSize()
```