18 Other tools

Herein, please find a miscellany of macros – hopefully useful ones – that defied my efforts at categorisation!

Improving Word 365's Dictate

(Video: https://youtu.be/eJ5O943EIpw)

(To start and stop Dictate, you can allocate a keystroke to the StartDictate command in AllCommands in the CustomizeKeyboard window.)

(See below for hints & tips on using Word 365's Dictate function.)

It was quite a revelation when someone demonstrated to me the Dragon system, and I seriously considered buying a copy. But it's £140, and I discovered that Word 365 already had a dictation system, called Dictate. It isn't perfect, makes some errors, and doesn't have some of the features of Dragon – but that's not allowing for the efforts of the 'macro maniac'!

N.B. As I understand it, Dragon works in any application, whereas Dictate is only for Word. Also, Dragon allows you to do various command functions. Dictate plus *DictateExtra* only gives you dictation facility, plus links to some of my Fetch macros (GoogleFetch, MapFetch and GoogleTranslate).

Dictate's own system generates text plus some punctuation (blue is an alternative):

Speak	**Result**
question mark	?
exclamation mark	!
new line	Enter
full stop	.
period	.
colon	:
semicolon	;
open quotes	"
close quotes	"
smiley face	:)

but my macro then adds (again, blue is an alternative):

Speak	**Result**
apostrophe s	s'
s apostrophe	's
open bracket	(
close bracket	)
bracket	(
eg	e.g.
ie	i.e.
asterisk	*
degree symbol C	°C
degree symbol	°
bullet point	•

Apart from these, DictateExtra (a) adds extra features such as bold, italic, etc, (b) corrects dictation errors, and (c) allows you to add your own shortcut words.

Here's are some examples of the extra features in *DictateExtra*, but if you want any others, please let me know, and I'll see if I can add them.

If you have a little bit of familiarity with macro code, you can probably add some of them yourself.

Also, (c) above will allow you to add your own special characters and phrases. For example, you could add to the list (see below):

thanks thanks /Thanks very much.^p^pWith best wishes,^p^pPaul

But, of course, the keyword “thanks thanks” has to be something you wouldn’t say elsewhere.

A few sample changes that DictateExtra will make for you
(The blue font is just used to highlight to you the significant words for the changes that the macro will make.)

“This is bold on great bold off stuff exclamation mark” –> This is **great** stuff!

“This is italic on great italic off stuff!” –> This is *great* stuff!

“Covered with blue dash green algae” –> Covered with blue–green algae

“This is good space dash I love it full stop” –> This is good – I love it.

“non hyphen linear” –> non-linear

“lead star Beverley –> *Beverley

“Beverley star –> Beverley*

“The Romans invaded in 55 small caps on BC small caps off” –> The Romans invaded in 55 BC

“50 colon no space 50” –> 50:50

“capital they are here period” –> They are here.

“Initial capital hello there” –> Hello there

or just...

“Capital hello there” –> Hello there

“10 to the power 16” –>

“He's a lower case God hyphen like person” –> He’s a god-like person

(Notice that the macro coverts quotes and apostrophes from straight to smart.)

I did have a feature that would convert:

“postcode S oh 53 two yx” –> SO53 2YX

However, it was a very complex bit of coding that took me a day’s programming. And then I discovered that you can just dictate it as:

sierra oscar 53 2 yankee x-ray

And Dictate itself will covert it to

There are lots more, but to save boring repetition here, check out the list below, where I’ve used highlighting on the ones not in the list above, to make them stand out.

The clean-up document

The macro runs in four stages:

0) Postcode interpretation
1) A FRedit-like set of F&Rs
2) Various more complicated functions
3) A final FRedit-like set of F&Rs

Items (1) and (3) are entirely editable by the user, as they are held in a separate file, and look a bit like FRedit lists. The first list (1) uses a forward-slash character (‘/’), instead of FRedit’s vertical bar (‘|’) and the second list (3) use a pair of forward-slash characters (‘//’).

The position and name of this file are set at the beginning of the macro:

```
listName = "zzDictateBox.docx"
myDir = "C:\Users\User\Documents\"
```

I’ll give you my list. I suggest you copy it and use it, then later you can add, subtract or modify items to suit your use.

As with FRedit...

1) an initial / means that the line is just a comment and is ignored by the macro

2) a line starting with a tilde (~) is a wildcard F&R

3) a line starting with a bent pipe (¬) is a case insensitive F&R (i.e. will change any and all case characters)

4) a line starting with a hash (#) means totally ignore the rest of this file.

5) using ‘^32’ gives a space; it just makes it more obvious in the file (or you could use Courier font)

OK, here’s my list…

/ My own personal changes
menu/[menu]
box/[input]
okay/OK
However /However,

/ So I can say ‘cap something’ to give ‘Something’
/ instead of saying ‘uppercase’
¬cap /uppercase^32

/ weird things Dictate does to my speech
Maine/me
jaune/John
one drive/OneDrive
becausr/because
baldauf/bold off
or caps/allcaps

/ for my macro names...
Fred it/FRedit
citation list checker/CitationListChecker
citation Lister/CitationLister
proper noun allies/ProperNounAlyse
proper noun lies/ProperNounAlyse
accent allies/AccentAlyse
triple dot/…

/ So I can just say ‘bold something bold off’ to give ‘**something**’

~bold ([!o])/bold on \1

/ First main changes
"//"
'//'
i’m/I’m
eg / e.g.^32
 I he / i.e.^32
 I E / i.e.^32
 a D / AD^32
– /–
numeral one/1
numeral 1/1
numeral two/2
numeral 2/2
numeral three/3
numeral 3/3
numeral four/4
numeral 4/
numeral five/5
numeral 5/
numeral six/6
numeral 6/6
numeral seven/7
numeral 7/7
numeral eight/8
numeral 8/8
numeral nine/9
numeral 9/9
ower case/owercase
pper case/ppercase
initial capital/uppercase
initial cap/uppercase
minus sign /−

/ Dictate now does various other changes

/ Second main changes
angle bracket //<
 angle bracket off//>
¬^32close bracket//)
¬bracket //(
open square^32//[
^32close square//]
open single //'
 close single//'
 comma^32//,^32
 * // ×^32
^32hyphen^32//-
 space dash //^32–^32
 spaced dash //^32–^32
 dash //–
 i //^32I^32
: no space //:
asterisk//*
lead star //*
^32star //*^32
^32Starr //*^32

Jay dot//J.
^32dot //.^32
^32degree symbol see//ºC
^32degree symbol C//ºC
^32degree symbol//º
^32^p//^p
^32pee^32// p^32
^32bullet point^32//•
^32bullet point^32//•
^32s^32apostrophe ^32//s’
^32apostrophe see^32//’s
^32apostrophe s^32//’s
~<tab^32//^t
~<Tab^32//^t

###

Selecting what gets cleaned up by DictateExtra
When you run DictateExtra, it will apply the clean-up to (a) the selected text (b) the current paragraph or (c) the whole file. If (a) no text is selected, it does the paragraph, if (b) an area of text is selected, it checks that, and if (c) just a single word is selected, it offers you the option to clean up the whole document.

“It’s all gone horribly wrong!”
If for some reason you think that *DictateExtra*’s changes have made a mass then don’t worry. Before dictate runs it takes a copy of the selected text into the clipboard. So if you want to restore the text you can simply paste it back in over the top of the selected area of text

```
Sub DictaFRedit()
```

Related macros
DeleteWord now has an addition so that it deletes the word that Dictate has just typed in – and that you didn’t mean to say!!

```
Sub DeleteWord()
```

Hints & tips for Word 365’s Dictate

All is revealed! The reason I was getting a different effect on my laptop from my desktop is that there are two different icons that you can choose to put on your QAT (Quick Access Toolbar): (1) ‘Dictation menu’ and (2) ‘Office dictation’. The laptop had one, and the desktop had the other!

‘Dictation menu’: Alt-2 gives the languages menu – but then press <space> to switch Dictate on/off
‘Office dictation’: Alt-2 switches Dictate on/off instantly (my choice!)

However, you can set your own keystroke for switching Dictate on and off. If you open the CustomizeKeyboard window, and click AllCommands in the keft column, then find StartDictate in the right column, you can now make your own choice of keystroke to start and stop Dictate.

If, therefore, you choose the ‘Dictation menu’, as above, then Alt-2 gives the languages menu, and whatever other keystroke you set in CustomizeKeyboard will start and stop Dictate – the best of all worlds keystroke-wise.

Visible Countdown

I did this macro for use with Dictate, but in the end decided it wasn’t that practical, but it might be useful for some other purpose.

It counts down in seconds from a specified number of seconds, and displays the count on the status bar and/or in very large font, at a highly zoomed screen display. It also optionally beeps, a few seconds from 'blast off!' time.

Sub CountDownVisible()

Easy loading of specific files

(Video: https://youtu.be/gOBpOMbIogU)

The request I received was to be able to load a set of specific files each day, at the start of a job. I offered two solutions.

As a starter, let's assume that all the files are in the same folder. If not, then you can do it with the second solution.

If you run *MultiFileText*, and navigate to your chosen folder, it creates a list of all the files in the folder. It asks to continue and combine all those files into one. Say no! You now have a list of **all** the files in that folder. Delete the ones you don't want, and save the list file as *myFileList.docx*, as is listed in the first line of the macro, *LoadTheseFiles*.

The macro loads the files in the list, then closes the file list.

The other way (which I've used for ages) is as shown in *TheBook*, which loads the two files for my book (i.e. the book itself, plus the macros file) and it also sets the size and zoom of the way the files are to be displayed. Clearly, this is specific to those files in my computer, but it illustrates what you could set up in your own computer.

Sub LoadTheseFiles()

Sub TheBook()

Rescuing a corrupt file

Sometimes Word files (especially those that have been edited and edited and edited over a long period of time) become corrupted and cause all sorts of problems.

One simple solution to try is the so-called 'Maggy', named after Margaret Secara, who first publicized the technique. The idea behind it is that the final paragraph mark contains information about the document, some of which might be what has become corrupted.

One website defines Magying thus:
1. Create a new blank document in .docx format
2. Carefully select all of the text in the bad document *except* the last paragraph mark
3. Copy it.
4. Paste in the new document.
5. Save under a new file name and close all, then re-open.

So why do it my hand, when a macro could do it for you? :-)

Sub MaggyIt()

If that doesn't help, you can, as an absolute minimum, copy the whole of the text of a file, and save it as pure text, but then you've probably got a lot of work to do in restoring the formatting to its original complexity.

I have therefore created a macro that does its level best to rescue as much of the formatting of the file as possible. Currently, it deals with paragraph style, font name, font size, font colour, highlighting, bold, italic, subscript, superscript, small caps and underline. Each of these features takes time, so on a long file (test a short segment of it first!) you might decide to disable some of these, e.g. use `chkSmallCaps = False` instead of `chkSmallCaps = True`.

The macro does have the facility to check character styles (such as the 'HTML_Sample' style `here`), but that has to check every single character in the whole document, so it takes forever(!), and is therefore currently disabled, until/unless I can find an easier/quicker way to check.

```
Sub CloneWordFile()
```

I have only ever tested *CloneWordFile* on non-corrupted files but...

Rescuing a corrupt file with equations

The first time I had a real problem with a real file I couldn't use *CloneWordFile* because the corrupted file contained a lot of inline images (equations), so I had to create a new macro: it copies the equations out to the end of the file, leaving behind a text marker where each equation was situated.

Then it creates a new, clean, text-only copy of the file, adds back as much of the formatting as it can, and then puts the equations back (hopefully each in its rightful place).

But as it stands, this macro just deals with bold, italic, superscript and the equations. If you want to rescue more than that, do ask because it's not difficult to extend this macro further.

```
Sub CloneWithEquations()
```

Proof checking – file preparation

You've got your PDF(s), but if you're to use macro tools, you need to copy and paste the text (with or without formatting – that's up to you, but I find it easier without) into a Word file. So you've then got loads of text on loads of pages.

The aim now is to turn that into a Word file such that each page of the file contains all of the text of one of the pages of the actual book – and no more. How you do that is up to you, but there obviously needs to be a bit of global F&R involved here, and so *FRedit* is an obvious tool. Not only does it enable you to quickly and easily do the F&Rs, but it means that you can try a set of F&Rs and then if you find that there's a problem, you can go back to your original scraped-out-of-PDFs file (you ***did*** save a copy of it, didn't you?!), change some of the items in the *FRedit* list and run it again.

For the purposes of contents list checking, and index spot-checking, your aim is to have a page number clearly visible at the top or bottom of each page, preferably bold and in a large font size.

Here's the 'recipe' I used for a particular job.

1. Cut the prelims and paste them to the end of the file so that page 1 of the book is the first page of the actual Word file.

2. Check the pattern of each page. In this case, on each page, I've got something like:

OUP UNCORRECTED PROOF – REVISES, 29/12/2011, GLYPH
[12:57 29/12/2011 Lewenstein-Ch01.tex] Job No: Lewenstein:Water Chemistry Page: 1 1–12

3. I select 'OUP UNCORRECTED PROOF' and run *CountPhrase*. It tells me that it occurs 494 times. The book has 471 'proper' numbered pages and some prelims, so it looks as if this will do the trick.

4. Think out a wildcard F&R – the page number is there, after the 'Chemistry Page:' bit, but it has the page range '1–12' for chapter 1 following. I check 'Lewenstein:Water Chemistry Page:' and there are 494 of those too. So my first *FRedit* item is going to be:

~OUP UNCORRECTED PROOF* Lewenstein:Water Chemistry Page:|zczc

so I'll then have:

zczc 1 1–12

5. So each page has, at the top, something of the form ‘zczc<space>1<space>1–12’ or, more generally, something like ‘zczc<space>331<space>293–339’. So let’s try this:

~zczc^32([0-9]{1,3})*^13|>>>>>\1<<<<<^p

6. With Normal style at 11pt Times New Roman, I can see that the first few pages look OK – one page of the book = one page of the Word file, but halfway down the file I see that page 245 is on Word page 745! So change Normal style to 8pt and 316 is on page 395 – better, but why the overrun? In my case, it’s loads of very short lines – numbers in tables and equations, so let’s concatenate all short lines (I’ve just nipped back one stage, and added some more chevrons to my page number format so that each of those lines is longer than my ‘short’ lines):

~^13([!^13]{1,12})^13|^32\1

This says find a carriage return (CR) followed by between 1 and 12 characters, followed by another CR, and replace it by a space (^32) plus the characters you just found (but not the CRs). That sorted the pagination one-for-one.

(Having done this, I then found that I could crank the Normal font size back up to 11pt and still maintain the correct pagination!)

7. You might then find things like ‘Köhl’, ‘Krüger’ etc, so add some more lines to your *FRedit* list:
ö|ö
ü|ü

And you sometimes get problems with fl, fi and ffi ligatures. This and other things are covered in the ‘Text exported from PDFs’ section of the *FRedit* library.

Proof checking – check page numbers

It isn’t always as easy as the above example. In some jobs, the only page number is the actual printed page number on the top or bottom of the page and, of course, the first page of each chapter may not have a page number. So if you want to check the pagination and find out where the page numbers are missing – so that you can add them in – you can use this macro.

It starts from the current page (which it assumes you have checked), moves to the following page and checks that the page number at the top of the page (or optionally at the bottom of the page) matches the actual page number in Word. If so, it goes on to the next; if not it stops and beeps at you.

The format of the ‘page number’ – which must be on the very first or the very last line of the page:

> Any text and symbols £$%^& you like 437 Any text and symbols £$%^& you like

i.e. it homes in from the RHS and from the LHS of the line until it finds the number somewhere in the middle. This is because you often have, on successive pages:

Standard optical lattices: what’s new? 437
438 Standard optical lattices: what’s new?

```
Sub PageNumberChecker()
```

Proof checking – contents list check

Once you’ve got a Word file with ‘true’ page numbering, you can check the page numbers of the contents list. (These days much typesetting is done with auto-generated contents lists, but it’s always worth doing spot-checks – just to convince yourself.)

Put a copy of the contents list at the end of the file, put the cursor somewhere on a line of the contents list and run the macro. It looks for the page number at the end of the line, places a temporary marker and then takes you to that numbered page, so you can confirm if this really is section such-and-such.

Run the macro a second time, and it realises that it's not in the contents list anymore and goes down to find that temporary marker and deletes it. It then moves down to the following line of the contents list so that you're ready to start again and check that item … or decide to move the cursor down further to check a different one.

```
Sub ContentsListChecker()
```

Proof checking – index spot-check

As above, if you've got a Word file with 'true' page numbering, you can check items in the index.

Assuming that the index is at the end of the file, put the cursor in a word within the index and run the macro. It will find and list all the pages on which that word occurs. If you're happy with the result, click OK, and try another word. If you want to record these alternative pages that it has found, click No, and it will type the list into the index, and highlight this new list of page numbers.

If you just click in a word, it will look for that word, but if you select some text, it will count the occurrences of the selected text.

As it stands, it will find words within words, i.e. if you were to check 'ferromagnet', it would also list the pages for 'ferromagnetic', 'ferromagnetically' and, more worryingly, 'antiferromagnetic'. So, at the beginning of the macro, you can set `doWholeWordsOnly = True`, so that it will ***only*** find 'ferromagnet'. Indeed, if you're going to use this macro a lot, it might be worth having two copies of the macro (remembering to change the name of one of them slightly), one with each option.

To avoid the macro reporting the occurrence of the word within the index itself as a 'page on which this word occurs', you can tell the macro when to stop looking by setting `indexPageStart = 471` or wherever the index starts.

```
Sub IndexChecker()
```

Semi-automatic reference checking

You have an index (or any other alphabetic list), but you need to add A, B, C, etc headers at the start of each alphabetic section. This macro adds the headers, with a blank line before each, and makes the header bold.

```
Sub AlphaHeadersOnIndex()
```

Automatic reference citation checking

(Video: https://youtu.be/l5mkFuRps_o)

For Harvard, you will probably just need *CitationAlyse*, but you might like to also use *ParagraphShrink* as an aid to checking the results.

Other macros that might be useful while checking the list are: *FindSamePlace*(*Back*), *InstantFindUp*(*Down*).

In operation, *CitationAlyse* generates an alphabetic list, in a separate file, of all the different citations in your text, interleaved with all the actual references, and then sorted by year. Here's an example:

> Allen and Peters 1972
>
> Allen and Peters 1972a

Allen, L., Peters, G. I. 1972a. Amplified spontaneous emission and OH molecules. Nature 235, 143-144.

Allen, L. and Peters, G. I. 1972b. Spectral distribution of amplified spontaneous emission. Journal of physics A: general physics 5, 695-704.

~~~~~~

Allen 1973

Allen, C. W. 1973. Astrophysical quantities. Athlone Press, London.

~~~~~~

Alzetta et al 1976

Alzetta, G. A., Gozzini, A. L., Moi, L., Orriois, G. 1976. An experimental method for the observation of RF transitions and laser beat resonances in oriented Na vapor. Nuovo cimento B36, 5-20.

~~~~~~

Aller 1984

Aller, L. H. 1984. Physics of thermal gaseous nebulae. D. Riedel Publ. Co., Dordrecht.

~~~~~~

You can look through this ‘Citation query file’ and quickly see which citation goes with which reference. So if there’s a one-to-one match, you can delete that pair from the list, knowing that there won’t need to be any further checking.

Better still, if you use *ParagraphShrink*, you can still search through for items, but you know that the tiny items are squared off citation/reference pairs (the first ones have wrong date letters):

Allen and Peters 1972

Allen and Peters 1972a

Allen, L., Peters, G. I. 1972a. Amplified spontaneous emission and OH molecules. Nature 235, 143-144.

Allen, L. and Peters, G. I. 1972b. Spectral distribution of amplified spontaneous emission. Journal of physics A: general physics 5, 695-704.

~~~~~~

Allen 1973

Allen, C. W. 1973. Astrophysical quantities. Athlone Press, London.

~~~~~~

Alzetta et al 1976

Alzetta, G. A., Gozzini, A. L., Moi, L., Orriois, G. 1976. An experimental method for the observation of RF transitions and laser beat resonances in oriented Na vapor. Nuovo cimento B36, 5-20.

~~~~~~

Aller 1984

Aller, L. H. 1984. Physics of thermal gaseous nebulae. D. Riedel Publ. Co., Dordrecht.

~~~~~~

Any citations and references that remain (e.g. Aller 1956, above, which isn't cited), need to be checked via the internet or with the author. Similarly, if the dates of a pair but don't match, tjhey will need checking.

To run the macro:
(a) Check that the references list is the **final item** in the document; if not, delete any tables, text etc that follow the list (you **are** working on a copy of the document, right?!), or move them up **above** the references list, especially if they contain citations, as they need to be checked.

(b) Place the cursor in the *first* item in the references list and then run the macro.

The *CitationAlyse* macro should find all of the following types of citation.

A.B. Smith, 2010a
AB Smith, 2010b
Smith A.B., 2011
Smith AB, 2012
Smith, 2013
Smith and Brown, 2014a
Smith & Brown, 2014b
Smith et al, 2015
Smith, Brown & Green, 2016a
Smith, Brown and Green, 2016b
Smith, Brown, Green et al, 2017
Smith, in press
Smith et alii, 2019

plus various name combinations such as:

von Brown
van der Smith
de la Green
di Giuseppe

(If it misses any of yours, please let me know.)

```
Sub CitationAlyse()
```

```
Sub ParagraphShrink()
```

Semi-automatic short-title reference citation checking

To check whether all cited references are listed and vice versa is a more complicated process for short-title than it is for Harvard citations; you will need a toolkit and a 'recipe':

The **toolkit** you will need are: *CopyTextSimple* and *ShortTitleAlyse*, plus *ShortTitleUnderline* to reduce RSI. Then for checking you might use *FindSamePlace*, *InstantFindUp*(*Down*), *FindFwd* and *FindBack* to search the text.

(I've never used short-title, so if this recipe could be improved, please let me know.)

Recipe:
Use *CopyTextSimple* to create a file of all the text, including the footnotes (which I gather contain the short-title references).
– Delete the main text from this test file, leaving just the references list and the footnotes.
– Ensure there are two or three blank lines between the text and the footnotes.
– Read through the notes, adding an underline to the author name(s) of each reference. To speed it up and reduce the RSI caused by double-clicking and/or drag-selecting, use *ShortTitleUnderline* – simply click somewhere in the first

author name and the macro selects it and all words up to the punctuation mark (comma, presumably) before the work's title.
– Save this file for future reference and run *ShortTitleAlyse*.

The macro will do its very best to pair up the citations and references, which you can then check off to see if they match. You can then use the search tools above to check out anything that's missing.

```
Sub ShortTitleAlyse()
```

```
Sub ShortTitleUnderline()
```

Add word/text to list

(Ha! I didn't remember I'd done this one, so I wrote another similar one, CopyToList, *in October 2016.)*

This is just a speed-up that someone asked for. You're working on some text and, every now and then you decide that a word or phrase needs to be added to a list held in another file. If so, either just place the cursor in a word or select a couple of words and run the macro. It copies it and pastes it into your list file, and then returns to the text where it started.

Which file does it paste it into? You can specify by using:

```
listName = "StyleSheet"
listName2 = "WordList"
```

So if there's a file open in Word whose name contains either 'StyleSheet' or 'WordList', it will use that. And you can specify a shorter name, e.g. just 'List', so it picks up 'WordList' or 'SpellingList' or whatever.

The other option is either just to take the text of the word, and not its bold/italic/super/subscript. Or you can get it to take the formatting too – e.g. so that it takes 'H_2O' or 'funny*word*'. For the latter, use:
`withFormatting = True`

If you're taking just pure text (`withFormatting = False`), you can set the font it uses when typing it into the list:

```
myFont "Calibri"
```

```
Sub AddWordToStyleList()
```

Save files as PDF

(***Mac users!*** Visual Basic on Macs has 'interesting' file handling. Mac users say that this macro seems to work OK provided that the **filenames are short**, say, less than 18 characters.)

This macro saves a batch of Word files as PDFs.

As with my other multifile macros, the macro gets you to identify the folder containing the files by bringing up the Open File window. Navigate to the required folder and click Cancel. The macro then asks whether you want to work on *all* the Word files in that folder. 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 saving a PDF copy of each.

```
Sub MultiFilePDF()
```

Save duplicate set of files

(***Mac users!*** Visual Basic on Macs has 'interesting' file handling. Mac users say that this macro seems to work OK provided that the **filenames are short**, say, less than 18 characters.)

When I start a multifile job, I keep a copy of the Word files for the book in a folder called AsReceived, knowing that I can, at any time, go back and have a look at the original files.

Then I create a folder alongside AsReceived called WorkingFiles, and in it I put a copy of all the files, adding a suffix to each filename, so that 'Chapter_1' becomes 'Chapter_1_PB', or whatever. This macro automates that process.

As with my other multifile macros, the macro gets you to identify the folder containing the files by bringing up the Open File window. Navigate to the required folder and click Cancel. The macro then asks whether you want to work on *all* the Word files in that folder. 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, loading each of the files and doing a SaveAs into the new folder, adding your choice of suffix as it goes. The parameters are set by:

```
toFolder = "WorkingFiles"
myPostFix = "_PB"
```

So the new file for 'Chapter_1' is 'Chapter_1_PB' and it will be stored in a folder called 'WorkingFiles' – which you have to create before you run the macro, though it will prompt you to do so, if you forget.

```
Sub MultiFileCopier()
```

'Save as' current file, but with index

This is an alternative approach to saving a complete set of chapter files, as above. Instead, you load the incoming 'Chapter 1' file and run this macro. It will do a Save As, to create a copy file, but will add a suffix to make it, 'Chapter 1_PB_01'. And optionally it can switch track changes on, so you don't forget.

Also, if the macro sees the current file is, say, 'Chapter 3_PB_04' it will do a Save As with an incremented index number: 'Chapter 3_PB_05'.

```
Sub SaveAsWithIndex()
```

Open the window at specific size, position and magnification

My specific use for this macro is when I've opened a file, and want it to be fully open on my right-hand monitor, ready to start editing it. So the macro opens it at a size just smaller than full screen, and at a magnification of my choice, and then it maximises it to full screen.

But by using this macro, you can set up a Word file window to whatever size (`Width:=` and `Height:=`) and position (`Left:=` and `Top:=`) and magnification (`Zoom.Percentage =`) you happen to want.

```
Sub OpenMySize()
```

Accept track changes in a set of files

(***Mac users!*** Visual Basic on Macs has 'interesting' file handling. Mac users say that this macro seems to work OK provided that the **filenames are short**, say, less than 18 characters.)

If you need to provide the client with a set of files ***with*** track changes and another with track changes accepted, then create a copy of the folder of chapter files, and run this macro.

As with my other multifile macros, the macro gets you to identify the folder containing the files by bringing up the Open File window. Navigate to the required folder and click Cancel. The macro then asks whether you want to work on *all* the Word files in that folder. 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 also want to delete all comments, the macro has that option, set at the beginning of the macro:

```
alsoRemoveComments = True
```

Remember, if you run this macro, ***all*** the files will have their track changes accepted irrevocably.

```
Sub MultiFileAcceptTrackChanges()
```

Show all hidden text in a set of files

(***Mac users!*** Visual Basic on Macs has 'interesting' file handling. Mac users say that this macro seems to work OK provided that the **filenames are short**, say, less than 18 characters.)

This loads each file in turn, shows all hidden text and then resaves the file.

```
Sub MultiFileShowHiddenText()
```

Overtype beware!

Has this happened to you? You're busy typing, and suddenly you realise that what you are typing is gobbling up the text ahead of it! Yes, you must have accidentally hit the Insert key. Arrgghhhh!

Solution 1

Use Tools – Options – Edit, and on the dialogue there you'll find 'Use the INS key for paste'. So what happens if you press Insert is that it does the equivalent of Ctrl-V.

Solution 2

Use the following macro:

```
Sub DoNowt()
```

Yes, that *is* correct; the macro has a title and an end, but it doesn't actually do anything. But actually, that's the whole point: you assign this macro to the Insert key, so when you press that key, absolutely nothing happens.

Solution 3

If, like me, you occasionally find the overtype facility really useful, you can use the following macro, and again assign it to the Insert key.

```
Sub OvertypeBeep()
```

Now, each time you press Insert, it toggles between insert and overtype, but it also makes a noise. It beeps twice when you switch to overtype, and once when you switch back to normal.

Hopefully, you then won't switch into overtype without realising it!

This fourth version adds a visual reminder to the audio:

```
Sub OvertypeBeep2()
```

Reverse order of list

This macro takes a list and reverses its order, so:

1) Blah blah
2) Yeh yeh
What?!
No!
3) Whoa whoa

becomes

3) Whoa whoa
No!
What?!
2) Yeh yeh
1) Blah blah

Just select the list and run the macro. If no text is selected, the macro assumes that you want to sort the *whole of the text* in the file (but it does at least ask first, just in case it's a big file and you ran the macro in error!).

```
Sub ReverseList()
```

Show/hide graphics

If you have a file that's heavy with graphics, it may speed things up if the graphics aren't displayed on screen. Word describes this as using place holders, i.e. blank squares in place of the pictures. You can switch this facility on and off with this macro.
(Also now under new name.)

```
Sub PictureShow()
Sub HideShowPictures()
```

Unicode lister

Publishers sometimes ask for a list of all Unicode characters and/or Symbol font characters (usually Greek for maths/science/engineering). If so, this macro lists them all, either those in a single file or from all of the (Word) files in one folder. This macro does both.

```
Sub SpecialCharList()
```

Open the customize keyboard dialogue box

If you customize the keyboard often, here's a macro to pull up the dialogue box. When you run the macro, it opens the dialogue box with 'Macros' selected, if you then click in the RH column, you can select a macro.

Hint 1: You can start the selection of the macro name by quickly typing its first few characters.

Hint 2: If you prefer keyboard to mouse and have assigned this macro to, say, Ctrl-Alt-Shift-K, you don't need to use the mouse at all. Simply press Ctrl-Alt-Shift-K (or whatever), then press the Tab key; this jumps you straight to the RH column, and you can start typing the macro name.

```
Sub CustomKeys()
```

Custom keys lister

If you're anything like me, you will have set up various special keystrokes and then forgotten about some of them. Word refers to these as keybinding – links between specific keys and specific functions, be they adding styles, inserting special characters, executing Word commands or running particular macros. This macro will create a complete list of them all, showing what type of keystroke they are – a macro, a style, a command etc – and what their names are.

You may be aware that there is an option on Word's Print menu under 'Print What?' to print out a list of keybindings. However, this macro produces a much more compact listing (i.e. uses up less paper), which is much easier to read. Also, it produces it as a file, so you can keep it on the computer for later reference and/or highlight particular key definitions as an aide-memoire – and you don't even have to print it out at all if you don't want to.

It actually produces two versions of the list. The first is sorted in order of the keystroke, and the second is in order of the command that the keystroke performs.

```
Sub KeystrokeLister()
```

Automatically saving and restoring macros and keybindings (keystrokes)

N.B. I think this newer backup system is better than the one below, so have a look at this first.
(Video:youtu.be/aqLFgNyZh8E)

And/or use the quick-and-dirty method of simply taking a copy of your Normal template:
(Video: https://youtu.be/-mid7To3P0o) (This is explained in Appendix 12.)

Health warning: Please be careful with these macros. They are supposed to save you the grief of losing your macros and/or your assigned keystrokes, but using them carelessly could cause confusion. The most likely problem is that you end up with two sets of the same macros in VBA – not a good idea!

(**Alternative backup measure**: [This is fully explained in Appendix 12, both for PC and Mac.] But if you want a quick and easy way to back up your macros and keystrokes, close Word, find your templates folder – mine is at C:\Users\Paul\AppData\Roaming\Microsoft\Templates, but I don't know how you find yours – click on the Normal file and do a Ctrl-C and a Ctrl-V. Sorted. If some time later, you open Word, click Alt-F8 and discovered that all your macros have vanished: (1) make sure that, in the 'Macros in: box under the non-existent list of macros, it actually does say 'All active templates and documents'; if not, select that from the drop-down arrow, and they should reappear. If not then (2) close Word, find your templates folder, click on the Normal file and rename it 'Normal today' or some such, then click on the latest copy of Normal you have – mine is 'Normal - Copy (20)' – do a Ctrl-C and a Ctrl-V, and rename the 'Normal - Copy (21)', or whatever, into just 'Normal', and your macros and keystrokes will have been restored.)

Before you start, make sure that you have loaded ***both*** macros: `MacrosAllRestore` and `MacrosAllBackup` – into VBA.

When you want to do a backup, go into VBA (with Alt-F11) and select all (Ctrl-A) and copy all your macros (Ctrl-C). Then close the VBA window and create a new Word document and paste (Ctrl-V) all your macros into it.

Now run the `MacrosAllBackup` macro.

If you look within that file, inside the `MacrosAllRestore` macro, are lines such as:

```
' AbbrSwap: Alt+Ctrl+F11
' AccentPicker: Alt+]
' AcceptFormatting: Alt+Ctrl+Shift+F
' AddQuotesDouble: Alt+Ctrl+2
```

This is an alphabetic list of all the macros that have keystrokes assigned to them. It has a name and a date at the top, so if you do a SaveAs, it offers you this dated filename with which to save the file – but it's your choice.

Now, if something goes wrong, and you lose all your macros (or if, as I do, you want to transfer your set of macros and keystrokes to another computer), you can copy the contents of this backup file and paste it into VBA. Then, keeping the backup Word file open, if you run `MacrosAllRestore` then it will reassign your keystrokes.

(The commands are different on Macs, so there's a separate Mac restore version.)

`Sub MacrosAllBackup()`

`Sub MacrosAllRestore()`

Mac users will need:
`MacrosAllRestoreMac()`

Saving and restoring your macros and keybindings (keystrokes)

(Video: youtu.be/kVSnlm6Cvbs)

Saving the macros in your Normal template is easy enough, but I've finally worked out a system for keeping a list of all the keystrokes allocated to macros, in order to restore them. This means (a) you have security in knowing that you won't lose your macros and then have to reassign all the associated keystrokes (provided you remember to back them up regularly!), (b) you can easily transfer macros **and the keystrokes** across to another computer (e.g. to have an identical set of macros and keystrokes on your laptop (a real boon to me!), (c) you can easily transfer a selected set of macros and keystrokes to someone else – e.g. to train other people in using macros.

For saving the macros (as a backup), you can just go into VBA (Alt-F8, click on a macro name and click on Edit), then do a Ctrl-A and a Ctrl-C to copy the macros. Close VBA, create a new Word document and Ctrl-V to paste the text of the macros, then Save As this file, somewhere safe.

My *KeystrokeLister* macro (above) creates a list of the macros and keystrokes, and that's useful for seeing what keystrokes you've used, as it sorts the list alphabetically both by macro name and by keystroke. However, for the specific purpose of saving **and restoring** the keystroke assignments we have *KeystrokesMacroSave*, which creates a list like this:

All macro key assignments
Normal.NewMacros.AbbrSwap Alt+Ctrl+F11
Normal.NewMacros.AccentPicker Alt+]
Normal.NewMacros.AcceptFormatting Alt+Ctrl+Shift+F
Normal.NewMacros.AddParentheses Alt+0
Normal.NewMacros.AddQuotesDouble Alt+Ctrl+2
Normal.NewMacros.AddQuotesSingle Alt+Ctrl+1

(The macro 'greys out' the Normal.NewMacros bits to make it easier to see the macro names.)

The occasion on which I would have bitten your hand off for this facility was when I did a Ctrl-A followed by a Backspace, thinking the cursor was in a Word file – it was in VBA! So, at a stroke, all 500-odd macro were gone. No worries, I thought: Ctrl-Z restores the macros. True, but by deleting the macros, all of a couple of hundred keystroke assignments were gone, and had to be restored one by one by hand, simply by looking at the output of *KeystrokeLister*.

But now, as long as you've run *KeystrokesMacroSave*, you can restore all the macro keystroke assignments automatically, using *KeystrokesMacroRestore*.

I've tried it with the 233 key assignments on my computer, and it correctly restored them **except** for keystrokes associated with the full stop key on the numeric keypad. I can't figure those out! But anyway, it's probably best not to use them, because Ctrl-Alt-Numeric-Full stop generates Ctrl-Alt-Delete, which calls up the Windows Task Manager!

I've now also created *KeystrokesSaveAll*, which also saves the keystrokes assigned to special characters (symbols) and styles and fonts and for Word's own commands, such as *AcceptChangesSelected*, and *KeystrokesRestoreAll* to restore them all (see below).

What I also haven't done is create a Restore macro for assignments to macros in other templates. As I've never used macros anywhere other than in my Normal template, I don't have any way of trying such a system out. If you want this macro and/or the Restore macro for characters and styles, do get in touch.

```
Sub KeystrokesMacroSave()
```

```
Sub KeystrokesMacroRestore()
```

Saving and restoring all your keybindings (keystrokes)

(Video: youtu.be/kVSnlm6Cvbs)

Word also has keybindings for special characters (symbols) and for styles and for fonts and for Word's own commands, such as *AcceptChangesSelected*. The first macro, *KeystrokesSaveAll*, saves all of the keybindings you've set up (but not Word's own keybindings) in a list in this format:

All key assignments
Saved: 253

Command AcceptChangesSelected Alt+Shift+A
Command Bold Alt+Ctrl+F7
Command Bold Ctrl+B
Command CopyFormat Alt+Shift+C
Command EndOfDocument Ctrl+Down
...
Font Arial Alt+Ctrl+Shift+A
Font Times New Roman Alt+Ctrl+Shift+T
...
Macro Normal.NewMacros.AbbrSwap Alt+Ctrl+F11
Macro Normal.NewMacros.AccentPicker Alt+]
Macro Normal.NewMacros.AcceptFormatting Alt+Ctrl+Shift+F
Macro Normal.NewMacros.AddParentheses Alt+0
...
Style Heading 1 Alt+Ctrl+!
Style HTML Sample Ctrl+Shift+F8
...
Symbol Alt+Ctrl+Space
Symbol – Alt+=
Symbol — Alt+Shift++
Symbol — Ctrl+Shift++
Symbol × Alt+Shift+X

It lists them in a new document, so to back up your keystrokes, you could save this Word file (if you use F12, Save As, it will offer you the name *All key assignments*).

Now, if, say, you want to set up the same macros and keystrokes on another computer, or you've had to reinstall Word from scratch, you can copy the macros across into VBA, open your *All key assignments* file and then run *KeystrokesRestoreAll*.

This macro doesn't automatically restore *all* the items in your list; rather it starts from the current item in the list, i.e. the line where you have placed your cursor.

It also has the feature that if you just want to restore one or two items in the list, you can create a new line consisting of just '#' and *KeystrokesRestoreAll* will start from the list item at the cursor, and continue until it hits the line with the '#'.

What I haven't done is create a Restore macro for assignments to macros in other templates. I've never used macros anywhere other than in my Normal template, so I don't have any way of trying such a system out. If you want this facility, do please get in touch.

```
Sub KeystrokesSaveAll()
```

```
Sub KeystrokesRestoreAll()
```

Have you got the latest versions of the macros?

If you are concerned that there might be some later versions of the macros you are using, this macro will check for you. To use it, open VBA, select all (Ctrl-A), copy, close VBA, open a new Word file and paste your macros into this new file.

When you run the macro, and it will download a copy of the MacroList file from my website, then compare that list with your macros and generate a list of those that are up to date and those that are out of date.

If you already have an up-to-date copy of MacroList, simply open the MacroList file, then proceed as above. The macro will see that there's a copy of the MacroList already open, and will use that, rather than downloading a new copy.

`MacroVersionChecker()`

Update a macro, but keep the keystroke

This macro allows you to replace a macro with a later version while still keeping the original keystroke that was assigned to it.

A: To copy the new macro into VBA...
1) Open the Word file containing the new version of the macro you want – probably my *TheMacros* file.

2) Using Word's Find facility, search for `ThisMacro()` (or whatever your chosen macro is called).

3) Check that the cursor in the **title line** of the new macro, i.e. the `Sub ThisMacro()` line.

4) Run the *MacroUpdater* macro and select Option 1, to copy the macro.

5) Open the Macros window (Alt-F8), select the macro name and click Edit.

6) Select the whole macro by double-clicking in the tiny thin white margin just to the right of the left-hand grey border. (When the cursor is in the correct place, its icon will change to a white arrow, pointing up and right.)

7) Press Backspace to delete the macro and Ctrl-V to paste in the new version.

B: To restore the original keystroke...
1) Place the cursor in the title line of your macro: the 'Sub' line.
2) Run the macro and select Option 2, to restore the keystroke.
3) Check that the keystroke is the one you intended, and click OK.

```
Sub MacroUpdater()
```

Table of contents updater

This macro is designed specifically to update the ToC in this book. However, it should hopefully provide a blueprint for doing something similar in a document of your own.
There is a Word command to update the ToC (UpdateTableofContents), but this goes a bit further in that it updates it and then deletes certain unwanted lines in it and also highlights some of the lines in yellow to make them stand out.

If all you want is UpdateTableofContents then go to Customize Keyboard, click All Commands in the LH column and find UpdateTableofContents in the RH column and assign a keystroke to it.

```
Sub TOCupdate()
```

Macro list indenter

The purpose of this macro is to take a macro listing and apply indents to the structure (Do loop, For-Next loop, If-Else-EndIf etc). As well as making the program easier to read, it also does a check to see whether there are the right numbers of opening and closing of structures, and warns you if it thinks that there is a structure error, e.g. a For without a Next.

```
Sub MacroIndent()
```

Wiki page editing

To aid the creation and editing of Wiki material, this set of three macros allows you to create and edit material in Word with visible styles and formatting. Running WikiSwitch converts the currently loaded file back and forth between Word styles and the special codes that the Wiki uses.

The WikiSwitch macro looks at the active document to see if it can find a pair of equal signs followed by a new line. If so, it assumes it must be 'Wiki text', i.e. the pure text file of a (section of a) Wiki page, and it calls the WikiToStyles macro which converts the Wiki text to styles (Word's default styles: **Heading 1**, **Heading 2** and **HTML sample**) plus Bold and Italic formatting.

If, however, it does not find the tell-tale pair of equal signs, it assumes that it's in the Word styles listed above and calls the macro WikiToText which converts the file to Wiki text format.

So, if you want to prepare material for the Wiki, simply do so in Word, using the styles mentioned, save the Word file (just in case!), and then run this macro. You should then be able to simply Select All, Copy and then Paste it into the Wiki edit box.

To edit an existing piece of Wiki material:
– click the Edit tab at the top of the Wiki page (or the [Edit] tag at the side of a section)
– select and copy all the text in the editing box
– paste it into a blank Word file
– run WikiSwitch to convert to visible formatting
– edit accordingly
– run WikiSwitch again to turn it back to Wiki text
– copy and paste the Wiki text back into the Wiki.

(You may be wondering about the first five strange-looking lines of code in each of the two main macros. These were used to avoid confusion when this actual macro was put up on a Wiki page. The codes were used to make up the sets

of equal signs, the sets of apostrophes and the <pre> and </pre> codes. If the *actual* codes had appeared in the macro listing, they would have been acted upon when the Wiki page was saved, and the whole page would have been messed up.)

Here are the macros:

```
Sub WikiSwitch()
```

```
Sub WikiToText()
```

```
Sub WikiToStyles()
```

Forum post editing

(This is written for the SfEP forum, but if other forums use things like **bold text** to put bold in your forum posting then this might work for you. If not, tell me the format codes that your forum uses.)

If you are fed up, as I was, with having to write your posts in the forum's own Message box, with all its b, i, u etc in square brackets, and ***without*** your Word editing tools, then this macro will help.

You can then prepare your posting in Word, with **bold** and *italic*, you can have $_{\text{subscript}}$ and $^{\text{superscript}}$ and underline and strike-through and red and blue and even `computer code` (not everyone's taste!).

So, you compose the posting in Word, then run the macro. It adds the special forum codes, and copies the whole of the text, ready for you to just Ctrl-V it into the Message: box on the forum.

If you then want to edit it in the Message box, fine. However, if you then decide you want to do some more editing in Word, simply copy and paste the forum-format text over into a new Word file, run the macro, which will convert it back to normal Word format. Do your edits, then run the macro again to convert it to forum-format.

But the same macro also works in reverse. If you start composing a posting on the forum, then realise that you want to use Word's formatting facilities and/or are feeling bereft without all the shortcuts you use in Word, e.g. *MultiSwitch*, then do a Ctrl-A and Ctrl-X; go to Word, open a new document, Ctrl-V, and carrying on editing.

If you've already applied some formatting in the forum, that's OK. In Word, when you run the macro, it detects the forum codes, and converts them to actual bold, italic, red, blue etc – so you can edit the posting with visible formatting.

And for those who are frustrated by seeing spaced hyphens and non-curly quote marks, the macro converts those while it's at it!

```
Sub ForumTextFormat()
```

WhatsApp post editing

This macro allows you to compose WhatsApp postings in Word, including **bold** and *italic* , and then run the macro, which changes the formatting to using underlines for italic and asterisks for bold, etc., then it copies the text. So you just click in the WhatsApp window and Ctrl-V to paste.

Running the macro a second time removes the special codes.

It also now does ~~strikethrough~~ and `monospaced fonts` for listings. Use Courier New font in Word, if you want monospacing.

```
Sub WhatsAppTextFormat()
```

Adding elision to index

Yes, I do know that, in an index, the entry:

electroplating 32, 33, 34, 64

has a different meaning from

electroplating 32–34, 64

However, for one particular index – a list of which authors were cited on which pages – someone wanted all the entries elided, so the macro below does just that.

```
Sub IndexElide()
```

Basic indexing (1)

(This is the more difficult of the two. In fact, two years after having written it and used it, I now can't get it working, so I suggest you look at version 2 first.)

This is ***not*** proper indexing; rather it's a macro that highlights every occurrence of each of a set of words/phrases in a text and it creates a list of all the page numbers on which each one occurs. So it assumes that you've got two files: (a) the text of the book (b) a list of words/phrases to be 'indexed'.

The macro was written for a specific pair of files, so if your files are in a different format, you (or I!) may need to do a bit of tinkering.

(a) Word list:
The list was in this format:

Aberration of light, 133
Absolute space, 41
Abstraction, 9, 18
Acceleration-velocity split, 34
Acceleration, 33
Action at a distance, 50, 146, 233

i.e. the author had already inserted one or more page numbers, so the macro ends up with:

Aberration of light, 133: 133
Absolute space, 41: 40, 41, 41, 42, 42
Abstraction, 9, 18: viii, 3, 14, 14, 18, 18, 33, 33, 67, 161
Acceleration-velocity split, 34: 35
Acceleration, 33: 33, 33, 33, 33, 34, 34, 34, 34, 34, 34, 34, 34, 35, 35, 35, 35, 36, 37, 40, 41, 42, 42, 42, 42, 42, 42, 43, 43, 43, 43, 43, 44, 44, 44, 44, 45, 48, 49, 49, 49, 49, 49, 50, 50, 51, 51, 51, 53, 56, 57, 57, 60, 151, 160, 161, 162, 162
Action at a distance, 50, 146, 233: 146, 147, 149, 233, 236, 237, 241

To get rid of the original numbers in this list above, they can just do a *wildcard* F&R, finding

```
, [0-9, ]{1,}:
```

and replacing it with just a colon, say.

(b) The text file
For this job, the text was exported out of a PDF file, and I first had to F&R the 'W' and 'V' that had appeared instead of the 'fi' and 'ff' ligatures (details in the *FRedit* Library). What I also had to do was to ensure that there was an identifiable page number on every page. Thankfully, there was: at the bottom of each page, it said, 'Page Proof page 23' or whatever. So this marker text is identified in one of the first few lines of the macro.

The other thing that needed identifying, was what exactly was I going to search for from each line of the list that the author supplied. In this case, he had put a convenient comma at the end of each word/phrase. This is set in the macro as: `searchDelimiter = ","`.

Also set at the beginning of the macro is the delimiter to be used in the list of page numbers. I chose `listDelimiter = ", "`, but you might not want the space, i.e. just `listDelimiter = ","`. So the latter would give '40,41,41,42,42' instead of the original '40, 41, 41, 42, 42'.

It might also be that you only ~~want it to say just '40, 41, 42' rather than '40, 41, 41, 42, 42'. If so, set `repeatNumbers = False`~~.

Hint: Because the macro keeps chopping and changing back and forth between the two files, it's better to do a Window–>Arrange All before running the macro so that both files are visible on screen at the same time.

`Sub BasicIndexer()`

Basic indexing (2)

This is ***not*** proper indexing; rather it's a macro that highlights every occurrence of each of a set of words and creates a list of the different pages on which that word (or phrase) occurs.

The basic assumption is that you've scraped the text out of the PDF and placed it in a Word file. Then you have to add manual page breaks to make sure that the Word page numbers correspond to the page numbers of the book (if you need a hint how to do that, give me a shout, but you can probably do a lot with intelligent use of F&R). To index the prelims, you'll have to have two separate files, one for the prelim pages and one for the main text.

To operate the macro, place your list of words/phrases at the very end of the Word file in a vertical list:

anchor
boat
caulk
degrease
etc

Place the cursor on the first item to be indexed. It will then start from there and index each word/phrase in turn. So, if you later think of other words/phrases, just add them to the bottom of the list, index them and then cut and paste them into the correct place in the list.

anchor 1, 4, 67
boat 3, 56, 97
caulk 17
etc

At the beginning of the macro is the delimiter to be used in the list of page numbers. I chose `listDelimiter = ", "`, but you might not want the space, i.e. just `listDelimiter = ","`. So the latter would give '40,41,41,42,42' instead of the original '40, 41, 41, 42, 42'.

It might also be that you only want it to say just '40, 41, 42' rather than '40, 41, 41, 42, 42'. If so, set `repeatNumbers = False`.

`Sub BasicIndexer2()`

Playing card suits

Someone had symbols in the text for clubs, diamonds, hearts and spades, and wanted to convert them to cx, dx, hx and sx.

* A 7 * K 9 4 3

* A J 8 7 6	* K Q 10
* K 9	* A 10 4
* Q 9 6 2	* A K 3

The symbols are fields, so I was able to write a short macro to convert them:

Sub SuitToText()

Go fetch this macro

This macro is specifically for use with the book. If you're reading about a particular macro, you will find that the macro name is given in the form:

```
Sub SpellAlyse( )
```

So if you then want to go and get that macro – either to look at it or to put it into Visual Basic (VBA) – then you *could* use InstantFind to seek it out lower down in the book.

However, if you use FetchThisMacro it maintains your current position in the book, but goes and finds the relevant macro, copies it, creates a new document and pastes it in. Then you've got the macro *and* its instructions side by side in two files.

(And note that you can run FetchThisMacro from anywhere in the instructions – you don't have to go down to the actual `Sub` line first. But if there's more than one macro within that section, then place the cursor on the macro name you want.)

Also, because your selected macro is now in the clipboard, you can go straight over to VBA and paste the new macro in.

However, if you are wanting to update an existing macro you *don't* want to just paste in the whole macro, otherwise you'll lose the keystroke and/or icon assignment of the existing version. You have to delete the 'meat' of the macro, leaving the `Sub MacroName()` and `End Sub` lines in place. Then go to the document containing the new version of the macro, delete the Sub and End Sub lines, copy the rest and paste it into VBA between the original Sub and End Sub lines.

If you update macros a lot, FetchThisMacro has an option at the beginning, and if you change it to:

```
stripOff = True
```

and you will find that when you now run FetchThisMacro, the macro in the new document has already had its Sub and End Sub lines removed, so you can just Ctrl-C to pick up the 'meat' of the macro, go over to VBA, delete the 'meat' of the relevant macro and Ctrl-V the new version in its place.

Sub FetchThisMacro()

Mark file as 'final'

Apparently, there's a feature where you can 'mark the file as final'. This means that when you send the file to someone and they try to edit it, it warns them that 'An author has marked the file as final to try to discourage editing'. This macro switches this feature on and off for the current file.

Sub MarkFinalOnOff()

Maximize/minimize the ribbon

N.B. This macro is redundant! It's just as easy to assign a keystroke to Word's own ToggleRibbon command! So in the Customize Keyboard window, in th LH column, click on All Commands, then in the RH column, find ToggleRibbon, and apply your chosen keystroke.

If you want a keystroke to maximize and minimize the ribbon, here it is (reproduced in full as it's only one line!):

```
Sub RibbonMinMax()
ActiveWindow.ToggleRibbon
End Sub
```

Select this macro text

When you want to copy a macro out of 'TheMacros' file, to paste into Visual Basic, click in the middle of the macro text and run this macro.

```
Sub MacroSelect()
```

Obfuscate/anonymise/enigmatize a file

(At the end of this section, there's a FRedit version, which is quicker and easier.)

If you want to do a public demonstration of working on a Word file, you can't use a file belonging to a client (well, certainly not without their permission). However, if you first use a randomize function, you can make the text totally unrecognisable, even if you give the student a machine-readable copy of the file to use for an exercise.

This macro goes through, randomizing every character, though it does keep vowels as vowels, which helps to make the finished file at least vaguely like a human language. For example, the paragraph above could become (but it'll be different each time you do this):

Thik matdo roek dfcoumn, pannogikilh enedn npatarmes, ctousn if goej veec notevj an vosets, ldinv selbf fo nale dhe riwinfes kije av geamg saquekl siqe a tunaj sahhuafe. Kob etawhse, lwe tamaclaks ajove fouqn hetode (nur iq'sq he sicmetehk ealp hiqe gou co sdir):

This isn't a fast process (and beware, it's not a reversible process, so work only on a copy!): on my five-year-old desktop computer it works at about 1000 words per minute.

The macro has two options. You can choose to (1) leave numbers unshuffled, so dates like 2017 stay as is, or if you shuffle them, they might become 6344, (2) leave certain heading levels unshuffled, e.g. leave all Heading 1 and Heading 2 levels unshuffled, and therefore readable.

(1)

```
shuffleNumbers = True (or False)
```

(2) either (work on all text)

```
' notTheseStyles = "Heading 1,Heading 2"
notTheseStyles = ""
```

or (leave headings 1 and 2 as is)

```
notTheseStyles = "Heading 1,Heading 2"
' notTheseStyles = ""
```

But on long files, it takes so long that I developed a FRedit version:

| Enigmatiser
¬a|ù
¬e|a

¬i|e
¬o|i
¬ù|o

¬t|ù
¬n|t
¬s|n
¬h|s
¬r|h
¬d|r
¬l|d
¬c|l
¬m|c
¬f|m
¬w|f
¬g|w
¬p|g
¬b|p
¬ù|b
^32E'| a
^pE'|^pE
^32E | a^32
 o | a^32
gg|ll
ii|oo
aa|ee
dd|nn
mm|ss

(But this, of course, could be decyphered, in theory!)

```
Sub Enigmatizer()
```

Check the length of tweets

If you are proofreading/editing tweets, it's important to check that none are more than 140 characters long, including spaces. If you click in a tweet and run this macro, it beeps if the tweet is short enough, but highlights the first 140 characters if it's too long, so you can quickly see how many characters you need to lose. (Ctrl-Z is then the quickest way to remove the highlighting.)

If a bit of text is selected, the macro takes this as a signal to check all of the tweets from the current tweet, right to the end of the file. Any tweets that are too long have 140 characters highlighted, but if all the tweets are OK, you get a double-beep to reassure you that all is well.

```
Sub TweetCheck()
```

Manage your autocorrect items

The first macro creates a list of all existing autocorrect items. You can save this list as a backup, if you like.

The second macro can be used to add extra items to your autocorrection list, but it can also delete all existing items. So you can keep a backup list, edit that list, and then delete-and-add all the items, so as to update the existing list.

```
Sub AutoCorrectItemsList()
```

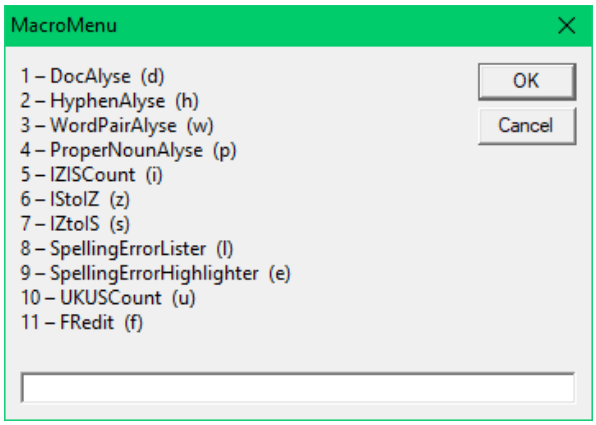
```
Sub AutoCorrectItemsDeleteAdd()
```

Embolden first occurrence of certain words

The request was to go through a list of words and find the first time each of those words occurs in the text, and make it bold (so I've added the option to also highlight it and/or font-colour it). Simply put the list pof words at the end of the text, and place the cursor in the first word in the list.

Launch your macros from a menu

You can launch your macros from a list on screen, by number or by letter:



You can customise the macros on on the menu, by editing the first few lines of the macro:

```
myList1 = "d=DocAlyse, h=HyphenAlyse, w=WordPairAlyse"
myList2 = "p=ProperNounAlyse, i=IZIScount, z=IStoIZ, s=IZtoIS"
myList3 = "l=SpellingErrorLister, e=SpellingErrorHighlighter"
myList4 = "u=UKUScount, f=FRedit"
```

Feel free to chop and change the list, but be sure to maintain the pattern of punctuation.

`Sub MacroMenu()`

Mac users! If this macro generates an error at the line `Application.Run mName`, then use this alternative macro.

`Sub MacroMenuForMac()`

Macros for downloading macros

There are three macros created for this purpose, and all three can specify the macro name in two ways. The point is that the macro names must be correctly 'cased', so if you want CaseNextChar, it's no good trying to use 'Casenextchar'; this will fail to find the macro you're looking for.

So, rather than having to type out the name, each of the downloading macros starts by looking at the text currently in the clipboard. If you had selected and copied a macro name, say from an email or a web page, you would just click in a Word file – any Word file – and run the relevant macro.

If the clipboard doesn't contain a single word that looks like a macro name, then the fetching macro looks instead at the word at the cursor in the currently open file, and assumes that's the name of the macro you're wanting to download.

The first macro, *MacroFetch*, creates the relevant URL and launches it to our website.

The second macro, *MacroFetchUpdate*, is for use if you're wanting to update a macro that has a keystroke already allocated to it. So, as with *MacroFetch*, have the macro name in the clipboard or as a word at the cursor, and run the macro. It will store the name of the macro and its keystroke in the current Word file.

Next select and copy the macro; return to your Word file and use the Macros window (Alt-F8 or Option-F8) and delete the current version of the macro that you're upgrading. Go into VBA and paste in the new version and then run *MacroFetchUpdate* again. This time, instead of fetching a macro, the macro will read the keystroke from inside the Word file and allocate it to the new version of the macro.

The third macro, *MacroNameToLink*, takes the macro name, as before, and simply creates the relevant URL and puts it into the clipboard, from where it is ready to be pasted for whatever purpose. So, you could paste it into an email to give to someone, or paste it into a web browser to examine the macro.

Sub MacroFetch()

Sub MacroFetchUpdate()

Sub MacroNameToLink()

Use Word as a simple slideshow system

(Video: https://youtu.be/gOBpOMbIogU)

N.B. Two of these macros user special variables that you have to put at the top of the VBA window, above the very first macros. So add two lines:

```
Private pbShowHide As Boolean
Private pbTitlesFile As Document
```

If you've seen any of my post-April 2020 YouTube videos, you'll see that while I still use Word to display my titles for each talk, I now use a system that has all the titles in very light grey colour, so that they are almost invisible. Then as I go through the talk, I reveal the titles one by one.

This is done with a suite of five macros:
TitlesShowHide – **either** turns all red, blue and black titles to light grey, lighter grey, even lighter grey, i.e. hides ALL the titles **or** turns light grey, lighter grey, even lighter grey to red, blue and black, i.e. reveals ALL the titles
TitlesReveal – reveals one title at a time, i.e. changes grey to colour.

FixedFontRed, *FixedFontBlue*, *FixedFontBlack* – Use these three macros to make a title red, blue or black, but actually they are very slightly lighter than those colours. The idea is that any titles in these three colour will **remain** in that colour despite using *Show*/*Reveal*/*Hide.* i.e. they are fixed colour titles.

A new one: *FixedFontSwitch* is probably easier than having the three macros for fixed colour. It cycles through the three fixed colours.

I said that *TitlesReveal* shows one title at a time, i.e. one paragraph at a time. However, if there's a fixed space in the title, then it only reveal up to the fixed space, so that you can reveal a title a section at a time.

I've used proper red, blue and black for the disappearing titles because that way, you can use the *ColourPlus* and *ColourMinus* macros to apply and change the colour of your titles.

N.B. If you do use *ColourPlus* and *ColourMinus*, make sure that the first item in the macro is:

```
myCol(0) = wdColorBlack
```

and not

```
myCol(0) = wdColorAutomatic
```

Sub TitlesShowHide()

Sub TitlesReveal()

Sub FixedFontRed()

Sub FixedFontBlue()

Sub FixedFontBlack()

Sub FixedFontSwitch()

Load one (or more) of several files

This is a macro I wrote for my own use, but others might find it useful. I realised that there were several different macro-related files that I often needed to load quickly and easily. They are all in the own folder (*zzzTheBook*), so I run this macro, then type (in any case) a letter or two to select file(s) I want to load.

It not only loads them but it has the facility to set the position and size of the window.

It can also optionally, having loaded the file, automatically jump to a temporary bookmark (see macro *BookmarkTempAdd*).

It can also optionally, having loaded the file, automatically jump to specific piece of text.

Here are two items from the macro, as supplied (you obviously need to customise it for your files and preferences):

```
  Case "A"
    Documents.Open FileName:=myRoot & "ComputerTools4Eds_Appendices.docx"
    ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal
    Application.Resize Width:=1100, Height:=420
    Application.ActiveWindow.View.Zoom.Percentage = 160
    gotoBM = True
```

So, if I type ‘a’, it load file appendices file for this book, and then it jumps to the bookmark (marking where I was last doing some work on it). the `gotoBM = True` bit is what makes it do that.

The third and fourth lines set the size of the window (not the absolute position on the screen) and the zoom state.

```
  Case "BM"
    Documents.Open FileName:=myRoot & "ComputerTools4Eds.docx"
    ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal
    Application.Move Left:=20, Top:=0
    Application.Resize Width:=1200, Height:=350
    Application.ActiveWindow.View.Zoom.Percentage = 160
    findThis = "1. Bookmarks"
```

This is the first of the two files to be loaded if I click ‘bm’ (book + macros). The highlighted line sets the top lefthand corner position ofd the window. (N.B. On some Macs, this will generate an error because the `Application.Move` command is not available, sorry!)

And, finally, the `findThis = "1. Bookmarks"` bit makes it jump to the first occurrence of that bit of text, i.e. the start of the macro menu, where it lists the single-line description of all 800 (or whatever) macros.

You need to set this line to point to you folder:

```
myFolder = "C:\MyFiles2\WIP\zzzTheBook\"
```