Hello! This is the tutorial to NMR Reporter!

This program requires you to have: Python 3.x (written in 3.7), [python-docx](https://python-docx.readthedocs.io/) package and

I created this program in an attempt to simplify the NMR Reporting process for articles, patents, etc. Some NMR processing software do not allow you to name your atoms as flexibly as one might want, let alone style those with subscripts, superscripts, italics, bolds and underlines. This is intending to fix this. Doing this in a .*docx* document is another matter, which is why I created this program.

The second reason I created this program is because sometimes, when you have to reuse some of your older material, it will have a wrong order of data, compared to the new journal where you want to submit. Fixing the order for each of the signals manually can be frustrating, especially if you have dozens of spectra to report. This program is intended to do this automatically in just a few seconds!

1. **Reading the spectra**

Before the spectra can be processed, they have to be set up correctly so that the program can parse them into something intelligible. To do that, you need to put your spectral data in one paragraph (in a *.docx* file, obviously), and have the previous paragraph start with the keyword Spectrum:. Note that this keyword is style-insensitive, but case-sensitive. After the keyword there must be a whitespace followed by the cypher for your spectrum. So, it must look something like this:

Spectrum: a

1H NMR (300 MHz, DMSO-*d*6) δ 0.86 (t, *J* = 6.7 Hz, 3H, 88), 1.21 – 1.34 (m, 10H, 83, 84, 85, 86, 87), 1.42 – 1.53 (m, 2H, 82), 1.93 (ddd, *J* = 13.2, 7.5, 6.0 Hz, 1H, 52’), 2.10 (ddd, *J* = 13.2, 6.0, 3.2 Hz, 1H, 52’’), 3.17 – 3.27 (m, 2H, 81), 3.52 (ddd, *J* = 11.8, 5.3, 4.1 Hz, 1H, 55’’), 3.57 (ddd, *J* = 11.8, 5.3, 3.8 Hz, 1H, 55’), 3.76 (ddd, *J* = 4.2, 3.7, 3.3 Hz, 1H, 54), 4.20 (ddt, *J* = 6.0, 4.2, 3.4 Hz, 1H, 53), 4.93 (t, *J* = 5.3 Hz, 1H, 5516), 5.16 (d, *J* = 4.2 Hz, 1H, 5316), 5.73 (d, *J* = 7.5 Hz, 1H, 5), 6.16 (dd, *J* = 7.5, 6.0 Hz, 1H, 51), 7.63 (t, *J* = 5.5 Hz, 1H, 414), 7.71 (d, *J* = 7.5 Hz, 1H, 6).

In this example, I gave my spectrum cypher ***a***, but it can be whatever you want. As of this moment there’s no restrictions against using the same cypher for as many spectra as you want, but I highly recommend against it! Also, if the first paragraph Spectrum: a is absent, the program won’t be able to see the spectrum itself. This means that each spectrum must be preceded by the Spectrum: keyword and an appropriate cypher.

Second, you must set up a format template. You need to do this because there’s too many ways in which spectra are reported, so instead of making the code guess what it’s reading, it’s best if you tell it what it should expect. Note that any mistakes in the format template and/or irregularity of the spectral data itself might result in an error!

First of all, there’re a few special notations in use:

%n – nuclide. Can have any value. In the example above that would be 1H.

%f – machine frequency. Can only accept integer or decimal numbers. In the example above that would be 300.

%s – solvent. Can have any value. In the example above that would be DMSO-*d*6.

These first three variables are specific to the ‘head’ of the spectrum report, then follow the signals, which may contain:

%m – multiplicity. Only a handful of values are accepted here (*s*, *d*, *t*, *q*, *p*, *x*, *h*, *b*, *r*, *m*, or *\** for a lopsided multiplet). In the first signal from the example above that would be t.

%j – *J* constant(s). Can only accept decimal numbers, a whitespace and a comma to separate several constants. Note that the ‘J = ‘ part and the ‘Hz’ part are not accepted as values of this variable! In the first signal from the example above that would be 6.7.

%i – integral. Can only accept an integer. In the first signal from the example above that would be 3.

%a – assignment. Can accept any value. In the first signal from the example above that would be 88.

In the end, a format template might look something like this:

%n NMR (%f MHz, %s) δ /%c (%m, \**J* = %j Hz, \*%iH, %a)/, /.

First, note the three slashes (‘/’). There must always be three slashes in your format template! Everything prior to the first slash is interpreted as the ‘head’ of the signal, like this:

‘%n NMR (%f MHz, %s) δ ‘

It contains general information about the spectrum as a whole. The first slash must always be preceded by a whitespace! The part between the first and second slashes is the signal format, like this:

%c (%m, \**J* = %j Hz, \*%iH, %a)

It is repeated as many times throughout the data as there are signals in it. Notice the ‘\*’ special notation. This denotes optional information that might or might not be there in your signal data. In the example above, some signals will be of the format:

%c (%m, *J* = %j Hz, %iH, %a)

While others will not contain the information about the *J* constants, and will be of the format:

%c (%m, %iH, %a)

‘\*’ symbols account for this. One such symbol opens an optional part, and the other closes. Hence, your format template must always contain an even number of ‘\*’! Also, note that it is desirable that each optional variable is contained within its own pair of ‘\*’s. This way, the program will search for those independently. For example, if some signals might not list the *J* constants, and others lack assignment(s), then the signal format might look something like this:

%c (%m, \**J* = %j Hz, \*\*%a, \*%iH) - for a signal that lists *multiplicity* first, then *J’s* (maybe), then *assignment(s)* (maybe), then *integral*.

The part between the second and third slashes is the delimiter. It is found between the symbols but not at the end of the spectral data. And lastly, your spectral data might contain a period at the end of the spectrum. That optional part goes after the third slash.

For the program to find your format template, you must precede it (the template) with a keyword Input format: and place it somewhere within your *.docx* file. Also, nothing else should be present in the same paragraph. The styling of the format template (or the keyword) is irrelevant. The format template can be listed anywhere in the document. Overall, it must look something like this (for the spectrum above):

Input format: %n NMR (%f MHz, %s) δ /%c (%m, \**J* = %j Hz, \*%iH, %a)/, /.

1. **Processing the spectra**

If the program is successful in both locating and reading the spectra, it will report you all the data for each signal of every spectrum it could find. This way you can easily check whether the program is working correctly and/or the spectra given contain no mistakes.

Currently, the program can only do two things: it can convert the format of a spectrum, and it can reassign the signals within each spectrum separately and/or globally. To tell the program which task it is to perform, you’ll need another line in your *.docx* file beginning with the keyword Tasks:, after which you state which task you want to perform: convert, or reassign. If you decide to do both, just put the two words one after the other separated by a whitespace with no comma and have no period at the end. So, it might look something like this (for just one task):

Tasks: convert

Or like this (for two tasks):

Tasks: convert reassign

The order in which tasks are listed is irrelevant. As always, the keyword is styling insensitive, but case-sensitive.

**2.1) Converting**

Converting in this context means using a different format template for the output than the one used in the input. This way you might change styling, variable order and/or notations. E.g., this:

Output format: **%n NMR (%f Megahertz, %s)** δ /%c (%iH, %m, \**J* = %j Hertz, \*%a)/, /.

contains some bold text, some changed notation (MHz -> Megahertz), and, most importantly, a different order of variables for each signal (*integral*, *multiplicity*, *J’s*, *assignment*, rather than *multiplicity*, *J’s*, *integral*, *assignment*). This feature allows you to reformat as many spectra as you want in a matter of seconds! To use this feature properly, your .*docx* file must contain a line opening with the ‘Output format: ‘ keyword (styling-insensitive and case-sensitive), followed by the desired format template (both styling- and case-sensitive). Note that in the current version of the program, the styling of the format template does not affect the styling of the variables in the spectra!

**2.2) Reassigning**

This feature is useful when you want to change the minimalistic assignments from whatever NMR program you’ve been working in to something more meaningful, or correct a mistake you made long ago in a large number of spectra. The way this works is you add a line opening with the keyword ‘Assignments: ’, followed by the cypher of the spectrum containing the assignment you want to change (this is the reason why it is strongly advised to use unique cyphers for each of your spectra). Then, list all the assignments in that spectrum that you want to change in a separate line, follow each of them by ‘ = ‘ and a new assignment. It must look something like this:

Assignments: p

281211, 281222 = (CH3)2((CH3)3C)Si-O-CH2-CH2-

28121, 28122 = 5′-OSi(C(CH3)3)(CH3)2

28411, 28422, 28433 = (CH3)2((CH3)3C)Si-O-CH2-CH2-

2841, 2842, 2843 = 5′-OSi(C(CH3)3)(CH3)2

65 = (CH3)2((CH3)3C)Si-O-CH2-CH2-

63, 64 = -CH2-O-CH2-CH2-O-CH2-

62, 66 = -CO3-CH2-CH2- + (CH3)2((CH3)3C)Si-O-CH2-CH2-

61 = -CO3-CH2-

3 = 3-NH

In case you want to make many similar reassignments at once, you should use ‘\*’ instead of an actual cypher. This will signal the program that the reassignment is to be performed on every spectrum it can find. Note that this non-specific reassignment does not restrict you to having a reassignment in *all* of the spectra: if the program cannot find a corresponding signal in the spectra, it will just move on.

Since the spectrum-specific reassigning is done first, it means you should not be worried there’ll be a conflict between the specific and non-specific reassigning: by the time non-specific reassigning commences, all the specific reassigning has been done and should not be further affected.

Note that the reassignment feature is both styling- and case-sensitive.

If you did not list ‘convert’ in your task, the program will use original input format to write down the new spectra. Otherwise, the output format template will be used.

1. **Executing the program**

Place the input .*docx* file in the same folder as the program. Execute ‘main.py’ file. You will be asked to give the input file’s name. This does not include the extension, which will always be .*docx*, so you must not add it there. Then the program will tell you how many spectra it could find, and will report every successful parsing of it. Then it will ask you to give it the name of the file where to write the output. As before, the extension should be omitted. Note that you cannot overwrite the input file. The resulting file will be written in Times New Roman, 14 Pts (this might be customizable in the future). Enjoy!

Also, feel free to check the examples to see how this program works!

If you have questions or have encountered a bug, feel free to DM me on reddit, u/Shevvv.