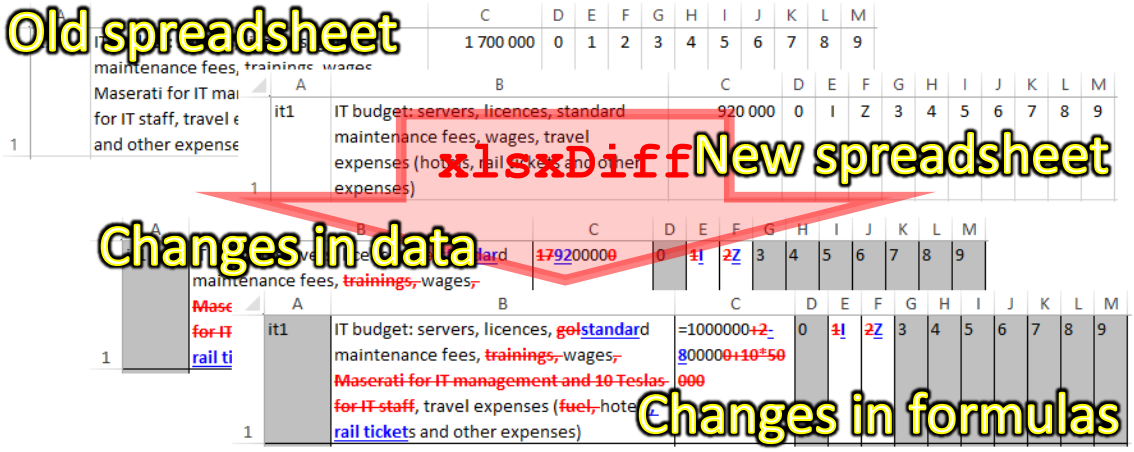


xlsxDiff

Python script for Excel spreadsheets comparison

Version 1.1.6

https://github.com/rafal-dot/xlsxDiff

Rafał Czeczótka

<rafal dot czeczotka at gmail dot com>

February 16th, 2023

1 Introduction 2
2 Installation 3
2.1 Python installation 3
2.2 Installation of OpenPyXL and XlsxWriter modules 3
3 Use 3
4 Options 4
4.1 “-f” “--formula” – compare formulas instead of data 4
4.2 “-x” “--highlight” – highlight columns and rows with changes 4
4.3 “-a” “--autofilter” – add automatic filter 5
4.4 “-e” “--noempty” – ignore empty cells 6
4.5 “-v” “--verbose” – verbose runtime output 6
4.6 “-q” “--quiet” – quite mode 6
4.7 “--version” – print version 6
5 FAQ 6
5.1 Does xlsxDiff have spreadsheet size limit? 6

5.2 In the output file, the error “#VALUE!” appears in some cells. How to fix it? 6
5.3 xlsxDiff shows that there are differences between cells, but no differences can be seen 6
5.4 A column/row/tab was added/removed between versions and an awful lot of differences appear, even though the changes were minor. What to do? 7
5.5 The script runs very slowly. Can I make it run faster? 8
5.6 What is PIP and how to find it? 8
6 Useful links 8
7 Changelog 9
8 Licence 9

1 Introduction

Excel is a powerful, complex and flexible tool. It is used for calculations, for storing data or for modelling complex interdependences. However you use it, you may find xlsxDiff useful. Especially if you work in a team and share data, you've surely encountered the challenge of identifying changes made by your workmates (or by yourself some time ago).

I myself have desired to compare two complex Excel spreadsheets many times. I was especially interested in finding things like minor modifications to texts in cells, modifications to numbers, or changes to formulas. Unfortunately, all the solutions I could find were limited to a simple binary comparison of cell values, which helps a lot, but is often too general and requires a huge extra effort to precisely identify changes made. Since I couldn't find a suitable solution, I finally got annoyed and wrote a solution myself which I am making available as open source.

The main purpose of this tool is to fill the gap and facilitate the search and visualization of changes made between file versions, with an emphasis on the ability to track changes made at the level of individual cells with visualization similar to changes tracking feature in Word. This script ignores all other changes made, like removing/adding/changing order of rows/columns/tabs, changes in formatting etc. However, it is easier to quickly identify where such general changes have been made and after minor manual interventions in the input files it is easy to get a comprehensive and clear picture of all changes made.

xlsxDiff uses two, widely used, but not part of any distribution I know of, Python modules. These modules allow the manipulation of Excel files: OpenPyXL and XlsxWriter.

xlsxDiff is designed to be used freely, without any obligation, in any environment, including commercial environment or large MNEs. xlsxDiff itself is released under the open-source GNU Affero GPL license, and I tried to make it based solely on tools and modules under open licenses (GNU Affero GPL, PSF License, MIT/Expat License and BSD 2-Clause License). However, just in case, consult your legal advisor. **And last but not least, remember that none of the licenses used provide any guarantees.**

2 Installation

2.1 Python installation

To avoid legal challenges, I suggest using the standard Python distribution, which can be found at https://www.python.org/downloads/windows/ [1] As of the date of this writing, the most current stable version is `python-3.10.10.exe`, but any version from 3.5 or above should be fine[2]. If you choose "Add python.exe to PATH" option during installation, it will make your life easier later.

2.2 Installation of OpenPyXL and XlsxWriter modules

Install two necessary modules being used by xlsxDiff, that allow to manipulate .xlsx files:

```
pip install openpyxl xlsxwriter
```

And *voilà*. That's it, you can enjoy using xlsxDiff.

3 Use

Using xlsxDiff is simple, in Windows environment just run `cmd` and call the script with three parameters: two input files and output file:

```
python xlsxDiff.py in1.xlsx in2.xlsx out.xlsx
```

It involves comparing two versions of a spreadsheet – the old one and the new one – resulting in a spreadsheet with all changes highlighted.

To make it easier to find the changes, colours are being widely used for marking tabs:

1. All changed tabs are standard (usually white) in colour;
2. All new tabs are coloured blue;
3. All deleted tabs are coloured red;
4. All tabs where no changes have been detected are grey in colour.

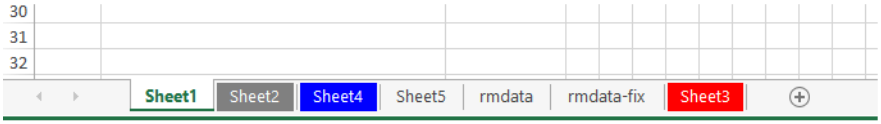


Figure 1 Tabs view: (i) sheets with white tabs contain cells compared item by item, (ii) grey is tab without any changes, (iii) blue tab is new one and (iv) red is removed tab

In the tabs where changes were detected (i.e. all except grey tabs):

1. Changed cells have a white background and in addition: unchanged text is black, **added text is blue and underlined** while ~~**deleted text is red and crossed out**~~;
2. In addition, when you select the "-x" option – to make it easier to find changes – in all rows where any changes are identified, the cell in the first column has a green background. Also, in all columns where any changes are identified, the cell in the first row has a green background. This allows you to easily filter the changed cells using Excel's built-in option to automatically filter by colour. Details are described in one of the following sections;
3. Unchanged cells have a grey background.

[1] For any Unix distribution you probably already have Python installed. I do not use macOS, but you can also find a distribution for this system

[2] The script uses some dictionary manipulation features introduced in version 3.5. For older version of Python 3 minor tuning might be required, what should be no problem for more advanced users. Please, RTFS for details 😉

	A	B	C
1	it1	IT budget: servers, licences, gold maintenance fees, trainings, wages, Maserati for IT management and 10 Teslas for IT staff, travel expenses (fuel, hotels and other expenses)	1 700 000

	A	B	C
1	it1	IT budget: servers, licences, standard maintenance fees, wages, travel expenses (hotels, rail tickets and other expenses)	920 000

	A	B	C
1	it1	IT budget: servers, licences, ~~gol~~standard maintenance fees, ~~trainings,~~ wages~~,~~ ~~Maserati for IT management and 10 Teslas for IT staff~~, travel expenses (~~fuel,~~ hotel~~s~~, rail tickets and other expenses)	~~1~~79200000

Figure 2 Example of compared cells: cells in the compared spreadsheets at the top and the result of the comparison at the bottom. Red text fragments were removed, blue text fragments were added, while the cell with the gray background was not changed

4 Options

4.1 "-f" "--formula" – compare formulas instead of data

By default, cell values are used to compare cells. These values were calculated by Excel when the spreadsheet was last used.

Use the "-f" option, if it is more important to compare changes in formulas rather than changes in data.

	A	B	C
1	it1	IT budget: servers, licences, gold maintenance fees, trainings, wages, Maserati for IT management and 10 Teslas for IT staff, travel expenses (fuel, hotels and other expenses)	1 700 000

	A	B	C
1	it1	IT budget: servers, licences, standard maintenance fees, wages, travel expenses (hotels, rail tickets and other expenses)	920 000

	A	B	C
1	it1	IT budget: servers, licences, ~~gol~~standard maintenance fees, ~~trainings,~~ wages~~,~~ ~~Maserati for IT management and 10 Teslas for IT staff~~, travel expenses (~~fuel,~~ hotel~~s~~, rail tickets and other expenses)	=1000000~~+2~~-800000~~+10*50000~~

Figure 3 "-f" comparison mode – compare formulas. See column C and compare with column C in the previous figure

4.2 "-x" "--highlight" – highlight columns and rows with changes

This parameter highlights rows and columns containing changes, making it easier to find them. In all rows where any changes are identified, the cell in the first column has a green background. Also, in all columns where any changes have been identified, the cell in the first row has a green background.

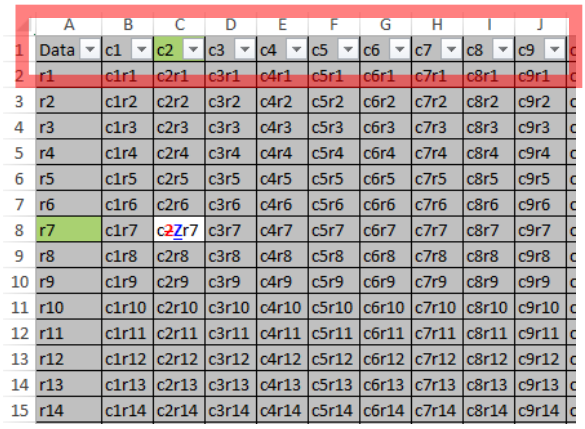
	A	B	C	D	E	F	G	H	I	J
1	Data	c1	c2	c3	c4	c5	c6	c7	c8	c9
2	r1	c1r1	c2r1	c3r1	c4r1	c5r1	c6r1	c7r1	c8r1	c9r1
3	r2	c1r2	c2r2	c3r2	c4r2	c5r2	c6r2	c7r2	c8r2	c9r2
4	r3	c1r3	c2r3	c3r3	c4r3	c5r3	c6r3	c7r3	c8r3	c9r3
5	r4	c1r4	c2r4	c3r4	c4r4	c5r4	c6r4	c7r4	c8r4	c9r4
6	r5	c1r5	c2r5	c3r5	c4r5	c5r5	c6r5	c7r5	c8r5	c9r5
7	r6	c1r6	c2r6	c3r6	c4r6	c5r6	c6r6	c7r6	c8r6	c9r6
8	r7	c1r7	c~~2~~Zr7	c3r7	c4r7	c5r7	c6r7	c7r7	c8r7	c9r7
9	r8	c1r8	c2r8	c3r8	c4r8	c5r8	c6r8	c7r8	c8r8	c9r8
10	r9	c1r9	c2r9	c3r9	c4r9	c5r9	c6r9	c7r9	c8r9	c9r9
11	r10	c1r10	c2r10	c3r10	c4r10	c5r10	c6r10	c7r10	c8r10	c9r10
12	r11	c1r11	c2r11	c3r11	c4r11	c5r11	c6r11	c7r11	c8r11	c9r11
13	r12	c1r12	c2r12	c3r12	c4r12	c5r12	c6r12	c7r12	c8r12	c9r12
14	r13	c1r13	c2r13	c3r13	c4r13	c5r13	c6r13	c7r13	c8r13	c9r13
15	r14	c1r14	c2r14	c3r14	c4r14	c5r14	c6r14	c7r14	c8r14	c9r14

Figure 4 Highlight columns and rows with changes

Note that Excel allows you to easily filter rows using Excel's built-in option to automatically filter by colour (see next section).

4.3 "-a" "--autofilter" – add automatic filter

This option causes an automatic filter to be added in all changed tabs in the first line automatically. Unfortunately, automatic pre-selection by colour is not available in the current version of the XlsxWriter library and manual intervention is required.



	A	B	C	D	E	F	G	H	I	J
1	Data	c1	c2	c3	c4	c5	c6	c7	c8	c9
2	r1	c1r1	c2r1	c3r1	c4r1	c5r1	c6r1	c7r1	c8r1	c9r1
3	r2	c1r2	c2r2	c3r2	c4r2	c5r2	c6r2	c7r2	c8r2	c9r2
4	r3	c1r3	c2r3	c3r3	c4r3	c5r3	c6r3	c7r3	c8r3	c9r3
5	r4	c1r4	c2r4	c3r4	c4r4	c5r4	c6r4	c7r4	c8r4	c9r4
6	r5	c1r5	c2r5	c3r5	c4r5	c5r5	c6r5	c7r5	c8r5	c9r5
7	r6	c1r6	c2r6	c3r6	c4r6	c5r6	c6r6	c7r6	c8r6	c9r6
8	r7	c1r7	c2r7	c3r7	c4r7	c5r7	c6r7	c7r7	c8r7	c9r7
9	r8	c1r8	c2r8	c3r8	c4r8	c5r8	c6r8	c7r8	c8r8	c9r8
10	r9	c1r9	c2r9	c3r9	c4r9	c5r9	c6r9	c7r9	c8r9	c9r9
11	r10	c1r10	c2r10	c3r10	c4r10	c5r10	c6r10	c7r10	c8r10	c9r10
12	r11	c1r11	c2r11	c3r11	c4r11	c5r11	c6r11	c7r11	c8r11	c9r11
13	r12	c1r12	c2r12	c3r12	c4r12	c5r12	c6r12	c7r12	c8r12	c9r12
14	r13	c1r13	c2r13	c3r13	c4r13	c5r13	c6r13	c7r13	c8r13	c9r13
15	r14	c1r14	c2r14	c3r14	c4r14	c5r14	c6r14	c7r14	c8r14	c9r14

Figure 5 Added automatic filters

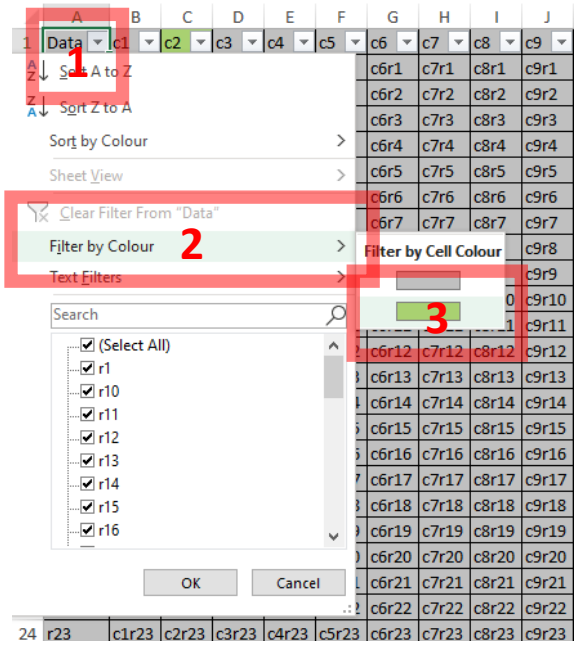
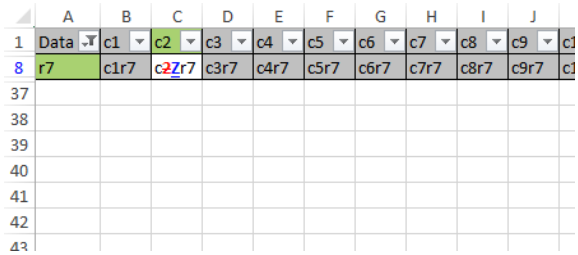


Figure 6 Steps to follow to preselect by colour: (1) expand the automatic filter menu in column A, (2) expand "Filter by Colour" menu item and (3) finally select green color



	A	B	C	D	E	F	G	H	I	J
1	Data	c1	c2	c3	c4	c5	c6	c7	c8	c9
8	r7	c1r7	c2Zr7	c3r7	c4r7	c5r7	c6r7	c7r7	c8r7	c9r7
37										
38										
39										
40										
41										
42										
43										

Figure 7 Result – only changed rows visible

4.4 "-e" "--noempty" – ignore empty cells

For sparse worksheets (i.e. worksheets with a small amount of data and a large number of empty cells), using this option can reduce the file size and increase processing speed. The disadvantage is that, for the same type of data, many not changed cells in our area of interest will not be marked with a grey background, which can make it more difficult to visually identify changes.

4.5 "-v" "--verbose" – verbose runtime output

Using this option increases the level of detail reporting at runtime. By default, xlsxDiff reports only the completion of a column comparison. In verbose mode, every cell comparison is reported, which can be important for very large spreadsheets to make sure the program is still working properly.

4.6 "-q" "--quiet" – quite mode

Disables all runtime messages. This option does not affect the messages generated by the modules used.

4.7 "--version" – print version

Prints version of xlsxDiff.

5 FAQ

5.1 Does xlsxDiff have spreadsheet size limit?

There are no size limits build in xlsxDiff. I have reports on of successful usage of the script with spreadsheets of hundreds of thousands of cells. Unfortunately, due the limits of OpenPyXL library, I have some reports about problems with spreadsheets with predefined names build in (nevertheless, this way of using Excel is not typical).

5.2 In the output file, the error "#VALUE!" appears in some cells. How to fix it?

xlsxDiff.py is just script that analyses texts and produces formatted output. It is as simple as that. Nothing more. Unfortunately such approach might cause unexpected errors, when Excel cannot properly interpret formulas in cells of output file. Fortunately, you can easily bypass this, just modifying content of cells. You can just replace = char with ' = chars (i.e. replacing single equals = char at the beginning of formula with two chars: apostrophe ' char and equals = char, what forces Excel not to interpret cell as formula, but as string).

5.3 xlsxDiff shows that there are differences between cells, but no differences can be seen

When displaying the contents of a cell, Excel trims spaces at the end of the cell's text, regardless of its formatting. So if the script shows that there are changes between cells and they are invisible in Excel, check if the cell contents end with spaces.

5.4 A column/row/tab was added/removed between versions and an awful lot of differences appear, even though the changes were minor. What to do?

xlsxDiff is a simple script that compares spreadsheets cell by cell. This tool does not allow to detect significant changes to the structure between versions. So if you think that catching such differences is important to you, even though this script doesn't do it, it allows you to easily identify where such changes were made. Then it may be prudent to manually add/modify the relevant rows/columns/tabs before starting the comparison:

1. if a column/row has been deleted, then add a corresponding empty column/row in 2nd file. As a result of the comparison, the content will be highlighted as deleted (see figures below),
2. if a column/row has been added, then add a corresponding empty column/row in 1st file. As a result of the comparison, the content will be highlighted as added (see figures below),
3. if the order of the columns/rows has been changed, you must unify the order between the input files,
4. if tab names have changed between file versions, it may be reasonable to unify the names so that the script can identify the appropriate pairs.

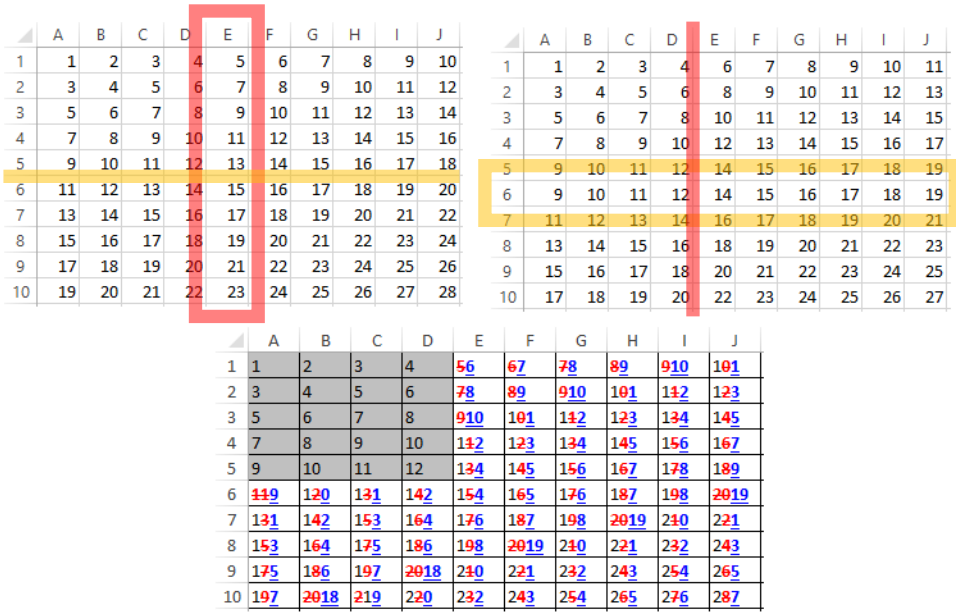


Figure 8 Original raw input data: row 6 added (orange) and column E removed (red). It is easy to identify where changes starts, but for following rows and columns the changes are unreadable

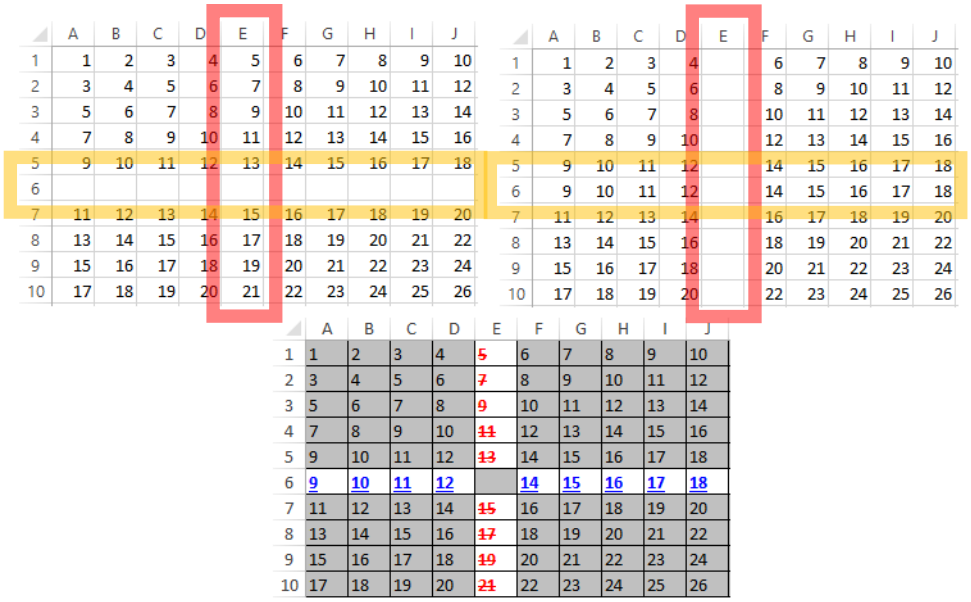


Figure 9 Artificially tailored input data: row 6 has been added again and column E has been removed. However, now, thanks to the synthetic changes made to the compared input spreadsheets (corresponding empty columns/rows have been added to 1st and 2nd spreadsheets accordingly), the comparison allows cell-by-cell identification where and what changes were made. Other parts of the spreadsheet, as expected, appear unchanged. The disadvantage of the solution, however, is the polluted addressing scheme

5.5 The script runs very slowly. Can I make it run faster?

Start by enabling the "--verbose" option. It is possible that the script detects data in the last rows/columns and performs a lot of unnecessary inspections. For example, using the list data validation function (see "Data" / "Data Tools" / "Data Validation" / "Validation criteria" in Excel), where a common solution is to store the source list at the end of the spreadsheet (somewhere around row 1,000,000). xlsxDiff is unable to detect that there are several hundred thousand empty cells between the end of data intended to be analysed and the validation data, and as a result performs millions of unnecessary operations. To speed up spreadsheet comparisons, it may make sense to manually interfere and reduce the size of the data to be analysed. A slight optimization of the spreadsheet and removal of redundant cells can result in significant increase of speed and increase of clarity of the output spreadsheet.

For more details see also the description of the verbose option as enabling such increased reporting makes it easier to identify aforementioned issue.

5.6 What is PIP and how to find it?

PIP is the "package installer for Python" and it is part of standard distribution. If the PIP program is not in the path, then you should look for pip.exe somewhere in the directory where you installed Python. By default, all Python files from the base distribution mentioned above are installed in the directory:

`C:\Users\<username>\AppData\Local\Programs\Python\Python310-32`

6 Useful links

Python 3 – a high-level, general-purpose programming language. See https://www.python.org/ PSF License;

difflib – Python module for comparing sequences, part of standard distribution. PSF License;

OpenPyXL – Excel files processing module. See https://openpyxl.readthedocs.io/ MIT/Expat License;

XlsxWriter – Excel files producing module. See https://xlsxwriter.readthedocs.io/ BSD 2-Clause License.

7 Changelog

Version	Date	Description
1.1	2023-02-11	Added option to ignore empty cells
1.0	2023-02-10	Initial version

8 Licence