latexindent.pl is a Perl script that indents .tex (and other) files according to an indentation scheme that the user can modify to suit their taste. Environments, including those with alignment delimiters (such as \texttt{tabular}), and commands, including those that can split braces and brackets across lines, are usually handled correctly by the script. Options for \texttt{verbatim}-like environments and commands, together with indentation after headings (such as \texttt{chapter}, \texttt{section}, etc) are also available. The script also has the ability to modify line breaks, and to add comment symbols and blank lines; furthermore, it permits string or regex-based substitutions. All user options are customisable via the switches and the YAML interface; you can find a quick start guide in Section 1.4 on page 5.

## Contents

1 Introduction .......................................................... 4
  1.1 Thanks ........................................................................ 4
  1.2 License ....................................................................... 4
  1.3 About this documentation ........................................... 4
  1.4 Quick start ............................................................... 5
  1.5 A word about regular expressions ............................... 6

2 Demonstration: before and after ........................................ 7

3 How to use the script ..................................................... 8
  3.1 From the command line ............................................... 8
  3.2 From arara .............................................................. 14
  3.3 Summary of exit codes ............................................... 14

4 indentconfig.yaml, local settings and the -y switch ............ 16
  4.1 indentconfig.yaml and .indentconfig.yaml .................. 16
  4.2 localSettings.yaml and friends ................................. 17
  4.3 The -y|yaml switch .................................................. 18
  4.4 Settings load order .................................................. 19

\* and contributors! See Section 11.2 on page 136. For all communication, please visit [\[\[13\]\]].
CONTENTS

5 defaultSettings.yaml .............................................. 20
  5.1 Backup and log file preferences ................................ 20
  5.2 Verbatim code blocks ........................................... 22
  5.3 filecontents and preamble ...................................... 25
  5.4 Indentation and horizontal space ............................... 26
  5.5 Aligning at delimiters .......................................... 26
    5.5.1 lookForAlignDelims: spacesBeforeAmpersand .............. 31
    5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash .......... 32
    5.5.3 lookForAlignDelims: the dontMeasure feature .............. 33
    5.5.4 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature . 35
    5.5.5 lookForAlignDelims: lookForChildCodeBlocks .............. 37
  5.6 Indent after items, specials and headings ....................... 38
  5.7 The code blocks known latexindent.pl .......................... 44
  5.8 noAdditionalIndent and indentRules ............................ 46
    5.8.1 Environments and their arguments ........................... 46
    5.8.2 Environments with items .................................... 53
    5.8.3 Commands with arguments .................................... 54
    5.8.4 ifelsefi code blocks ....................................... 56
    5.8.5 specialBeginEnd code blocks ................................ 57
    5.8.6 afterHeading code blocks .................................... 58
    5.8.7 The remaining code blocks ................................... 60
      5.8.7.1 keyEqualsValuesBracesBrackets .......................... 60
      5.8.7.2 namedGroupingBracesBrackets ............................ 61
      5.8.7.3 UnNamedGroupingBracesBrackets ........................... 61
      5.8.7.4 filecontents ............................................ 62
    5.8.8 Summary ................................................... 62
  5.9 Commands and the strings between their arguments ............... 62

6 The -m (modifylinebreaks) switch ................................ 68
  6.1 Text Wrapping .................................................. 70
    6.1.1 Text wrap quick start ....................................... 70
    6.1.2 textWrapOptions: modifying line breaks by text wrapping 70
    6.1.3 Text wrapping on a per-code-block basis .................... 73
  6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs . 78
  6.3 Combining removeParagraphLineBreaks and textWrapOptions ...... 84
    6.3.1 text wrapping beforeFindingChildCodeBlocks ............... 85
  6.4 Summary of text wrapping ....................................... 87
  6.5 oneSentencePerLine: modifying line breaks for sentences .... 88
    6.5.1 sentencesFollow ............................................ 90
    6.5.2 sentencesBeginWith ........................................ 90
    6.5.3 sentencesEndWith .......................................... 91
    6.5.4 Features of the oneSentencePerLine routine ................ 93
    6.5.5 Text wrapping and indenting sentences ...................... 94
  6.6 Poly-switches .................................................. 96
    6.6.1 Poly-switches for environments .............................. 96
      6.6.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-Line .... 97
      6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWith-LineBreak . 99
      6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary .......... 100
      6.6.1.4 Removing line breaks (poly-switches set to −1) .................. 101
      6.6.1.5 About trailing horizontal space .......................... 102
      6.6.1.6 poly-switch line break removal and blank lines .................. 102
    6.6.2 Poly-switches for double back slash ........................ 104
      6.6.2.1 Double back slash starts on own line ..................... 104
      6.6.2.2 Double back slash finishes with line break ............... 105
      6.6.2.3 Double back slash poly-switches for specialBeginEnd ............. 105
      6.6.2.4 Double back slash poly-switches for optional and mandatory arguments 106
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6.6.2.5 Double back slash optional square brackets</td>
</tr>
<tr>
<td></td>
<td>6.6.3 Poly-switches for other code blocks</td>
</tr>
<tr>
<td></td>
<td>6.6.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches</td>
</tr>
<tr>
<td></td>
<td>6.6.5 Conflicting poly-switches: sequential code blocks</td>
</tr>
<tr>
<td></td>
<td>6.6.6 Conflicting poly-switches: nested code blocks</td>
</tr>
<tr>
<td>7</td>
<td>The -r, -rv and -rr switches</td>
</tr>
<tr>
<td></td>
<td>7.1 Introduction to replacements</td>
</tr>
<tr>
<td></td>
<td>7.2 The two types of replacements</td>
</tr>
<tr>
<td></td>
<td>7.3 Examples of replacements</td>
</tr>
<tr>
<td></td>
<td>7 The -r, -rv and -rr switches</td>
</tr>
<tr>
<td></td>
<td>8 The -lines switch</td>
</tr>
<tr>
<td>9</td>
<td>Fine tuning</td>
</tr>
<tr>
<td>10</td>
<td>Conclusions and known limitations</td>
</tr>
<tr>
<td>11</td>
<td>References</td>
</tr>
<tr>
<td></td>
<td>11.1 External links</td>
</tr>
<tr>
<td></td>
<td>11.2 Contributors</td>
</tr>
<tr>
<td>A</td>
<td>Required Perl modules</td>
</tr>
<tr>
<td></td>
<td>A.1 Module installer script</td>
</tr>
<tr>
<td></td>
<td>A.2 Manually installing modules</td>
</tr>
<tr>
<td></td>
<td>A.2.1 Linux</td>
</tr>
<tr>
<td></td>
<td>A.2.1.1 perlbrew</td>
</tr>
<tr>
<td></td>
<td>A.2.1.2 Ubuntu/Debian</td>
</tr>
<tr>
<td></td>
<td>A.2.1.3 Ubuntu: using the texlive from apt-get</td>
</tr>
<tr>
<td></td>
<td>A.2.1.4 Alpine</td>
</tr>
<tr>
<td></td>
<td>A.2.2 Mac</td>
</tr>
<tr>
<td></td>
<td>A.2.3 Windows</td>
</tr>
<tr>
<td>B</td>
<td>Updating the path variable</td>
</tr>
<tr>
<td></td>
<td>B.1 Add to path for Linux</td>
</tr>
<tr>
<td></td>
<td>B.2 Add to path for Windows</td>
</tr>
<tr>
<td>C</td>
<td>latexindent-yaml-schema.json</td>
</tr>
<tr>
<td></td>
<td>C.1 VSCode demonstration</td>
</tr>
<tr>
<td>D</td>
<td>Using conda</td>
</tr>
<tr>
<td>E</td>
<td>logFilePreferences</td>
</tr>
<tr>
<td>F</td>
<td>Encoding indentconfig.yaml</td>
</tr>
<tr>
<td>G</td>
<td>dos2unix linebreak adjustment</td>
</tr>
<tr>
<td>H</td>
<td>Differences from Version 2.2 to 3.0</td>
</tr>
<tr>
<td>10</td>
<td>Conclusions and known limitations</td>
</tr>
<tr>
<td></td>
<td>List of listings</td>
</tr>
<tr>
<td>11.1</td>
<td>External links</td>
</tr>
<tr>
<td></td>
<td>Contributors</td>
</tr>
<tr>
<td>A</td>
<td>Required Perl modules</td>
</tr>
<tr>
<td></td>
<td>A.1 Module installer script</td>
</tr>
<tr>
<td></td>
<td>A.2 Manually installing modules</td>
</tr>
<tr>
<td></td>
<td>A.2.1 Linux</td>
</tr>
<tr>
<td></td>
<td>A.2.1.1 perlbrew</td>
</tr>
<tr>
<td></td>
<td>A.2.1.2 Ubuntu/Debian</td>
</tr>
<tr>
<td></td>
<td>A.2.1.3 Ubuntu: using the texlive from apt-get</td>
</tr>
<tr>
<td></td>
<td>A.2.1.4 Alpine</td>
</tr>
<tr>
<td></td>
<td>A.2.2 Mac</td>
</tr>
<tr>
<td></td>
<td>A.2.3 Windows</td>
</tr>
<tr>
<td>B</td>
<td>Updating the path variable</td>
</tr>
<tr>
<td></td>
<td>B.1 Add to path for Linux</td>
</tr>
<tr>
<td></td>
<td>B.2 Add to path for Windows</td>
</tr>
<tr>
<td>C</td>
<td>latexindent-yaml-schema.json</td>
</tr>
<tr>
<td></td>
<td>C.1 VSCode demonstration</td>
</tr>
<tr>
<td>D</td>
<td>Using conda</td>
</tr>
<tr>
<td>E</td>
<td>logFilePreferences</td>
</tr>
<tr>
<td>F</td>
<td>Encoding indentconfig.yaml</td>
</tr>
<tr>
<td>G</td>
<td>dos2unix linebreak adjustment</td>
</tr>
<tr>
<td>H</td>
<td>Differences from Version 2.2 to 3.0</td>
</tr>
<tr>
<td>List of listings</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
</tr>
</tbody>
</table>
SECTION 1

Introduction

1.1 Thanks

I first created latexindent.pl to help me format chapter files in a big project. After I blogged about it on the \TeX stack exchange [1] I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar [18] who helped to develop and test the initial versions of the script.

The YAML-based interface of latexindent.pl was inspired by the wonderful arara tool; any similarities are deliberate, and I hope that it is perceived as the compliment that it is. Thank you to Paulo Cereda and the team for releasing this awesome tool; I initially worried that I was going to have to make a GUI for latexindent.pl, but the release of arara has meant there is no need.

There have been several contributors to the project so far (and hopefully more in the future!); thank you very much to the people detailed in Section 11.2 on page 136 for their valued contributions, and thank you to those who report bugs and request features at [13].

1.2 License

latexindent.pl is free and open source, and it always will be; it is released under the GNU General Public License v3.0.

Before you start using it on any important files, bear in mind that latexindent.pl has the option to overwrite your .tex files. It will always make at least one backup (you can choose how many it makes, see page 21) but you should still be careful when using it. The script has been tested on many files, but there are some known limitations (see Section 10). You, the user, are responsible for ensuring that you maintain backups of your files before running latexindent.pl on them. I think it is important at this stage to restate an important part of the license here:

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

There is certainly no malicious intent in releasing this script, and I do hope that it works as you expect it to; if it does not, please first of all make sure that you have the correct settings, and then feel free to let me know at [13] with a complete minimum working example as I would like to improve the code as much as possible.

Warning!

Before you try the script on anything important (like your thesis), test it out on the sample files in the test-case directory [13].

If you have used any version 2.* of latexindent.pl, there are a few changes to the interface; see appendix H on page 148 and the comments throughout this document for details.

1.3 About this documentation

As you read through this documentation, you will see many listings; in this version of the documentation, there are a total of 560. This may seem a lot, but I deem it necessary in presenting the various different options of latexindent.pl and the associated output that they are capable of producing.

The different listings are presented using different styles:
1.4 Quick start

If you’d like to get started with `latexindent.pl` then simply type

```
$ latexindent.pl myfile.tex
```

from the command line. If you receive an error message such as that given in Listing 5, then you need to install the missing perl modules.
1.5 A word about regular expressions

As you read this documentation, you may encounter the term *regular expressions*. I’ve tried to write this documentation in such a way so as to allow you to engage with them or not, as you prefer. This documentation is not designed to be a guide to regular expressions, and if you’d like to read about them, I recommend [12].
SECTION 2

Demonstration: before and after

Let’s give a demonstration of some before and after code – after all, you probably won’t want to try the script if you don’t much like the results. You might also like to watch the video demonstration I made on youtube [30]

As you look at Listings 6 to 11, remember that latexindent.pl is just following its rules, and there is nothing particular about these code snippets. All of the rules can be modified so that you can personalise your indentation scheme.

In each of the samples given in Listings 6 to 11 the ‘before’ case is a ‘worst case scenario’ with no effort to make indentation. The ‘after’ result would be the same, regardless of the leading white space at the beginning of each line which is stripped by latexindent.pl (unless a verbatim-like environment or noIndentBlock is specified – more on this in Section 5).

<table>
<thead>
<tr>
<th>LISTING 6: filecontents1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{filecontents}{mybib.bib}</td>
</tr>
<tr>
<td>@online{strawberryperl, title=&quot;Strawberry Perl&quot;, url=&quot;<a href="http://strawberryperl.com/%22%7D">http://strawberryperl.com/&quot;}</a></td>
</tr>
<tr>
<td>@online{cmhblog, title=&quot;A Perl script ... url=&quot;...&quot;}</td>
</tr>
<tr>
<td>\end{filecontents}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 7: filecontents1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{filecontents}{mybib.bib}</td>
</tr>
<tr>
<td>@online{strawberryperl, title=&quot;Strawberry Perl&quot;, url=&quot;<a href="http://strawberryperl.com/%22%7D">http://strawberryperl.com/&quot;}</a></td>
</tr>
<tr>
<td>@online{cmhblog, title=&quot;A Perl script ... url=&quot;...&quot;}</td>
</tr>
<tr>
<td>\end{filecontents}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 8: tikzset.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tikzset{ shrink inner sep/.code={ \pgfkeysgetvalue... \pgfkeysgetvalue... } }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 9: tikzset.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tikzset{ shrink inner sep/.code={ \pgfkeysgetvalue... \pgfkeysgetvalue... } }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 10: pstricks.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\def\Picture#1{% \def\stripH[#1]{% \begin{pspicture}[showgrid] \psforeach{\row}{{3,2,8,2.7,3.3.1},% {2.8,1,1.2,2,3},% ... }{% \expandafter... } \end{pspicture}}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 11: pstricks.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\def\Picture#1{% \def\stripH[#1]{% \begin{pspicture}[showgrid] \psforeach{\row}{{3,2,8,2.7,3.3.1},% {2.8,1,1.2,2,3},% ... }{% \expandafter... } \end{pspicture}}</td>
</tr>
</tbody>
</table>
SECTION 3

How to use the script

`latexindent.pl` ships as part of the `TeXLive` distribution for Linux and Mac users; `latexindent.exe` ships as part of the `TeXLive` and MiKTeX distributions for Windows users. These files are also available from github [13] should you wish to use them without a `TeX` distribution; in this case, you may like to read appendix B on page 141 which details how the path variable can be updated.

In what follows, we will always refer to `latexindent.pl`, but depending on your operating system and preference, you might substitute `latexindent.exe` or simply `latexindent`.

There are two ways to use `latexindent.pl`: from the command line, and using arara; we discuss these in Section 3.1 and Section 3.2 respectively. We will discuss how to change the settings and behaviour of the script in Section 5 on page 20.

`latexindent.pl` ships with `latexindent.exe` for Windows users, so that you can use the script with or without a Perl distribution. If you plan to use `latexindent.pl` (i.e., the original Perl script) then you will need a few standard Perl modules – see appendix A on page 138 for details; in particular, note that a module installer helper script is shipped with `latexindent.pl`.

3.1 From the command line

`latexindent.pl` has a number of different switches/flags/options, which can be combined in any way that you like, either in short or long form as detailed below. `latexindent.pl` produces a `.log` file, `indent.log`, every time it is run; the name of the log file can be customised, but we will refer to the log file as `indent.log` throughout this document. There is a base of information that is written to `indent.log`, but other additional information will be written depending on which of the following options are used.

- **-v, --version**

```bash
cmh:~$ latexindent.pl -v
```

This will output only the version number to the terminal.

- **-h, --help**

```bash
cmh:~$ latexindent.pl -h
```

As above this will output a welcome message to the terminal, including the version number and available options.

```bash
cmh:~$ latexindent.pl myfile.tex
```

This will operate on `myfile.tex`, but will simply output to your terminal; `myfile.tex` will not be changed by `latexindent.pl` in any way using this command.

- **-w, --overwrite**
3.1 From the command line

```
cmh:~$ latexindent.pl -w myfile.tex
cmh:~$ latexindent.pl --overwrite myfile.tex  
cmh:~$ latexindent.pl myfile.tex --overwrite
```

This will overwrite myfile.tex, but it will make a copy of myfile.tex first. You can control the name of the extension (default is .bak), and how many different backups are made – more on this in Section 5, and in particular see backupExtension and onlyOneBackUp.

Note that if latexindent.pl can not create the backup, then it will exit without touching your original file; an error message will be given asking you to check the permissions of the backup file.

```
cmh:~$ latexindent.pl -o=output.tex,--outputfile=output.tex
```

This will indent myfile.tex and output it to output.tex, overwriting it (output.tex) if it already exists'. Note that if latexindent.pl is called with both the -w and -o switches, then -w will be ignored and -o will take priority (this seems safer than the other way round).

Note that using -o as above is equivalent to using

```
cmh:~$ latexindent.pl myfile.tex > output.tex
```

You can call the -o switch with the name of the output file without an extension; in this case, latexindent.pl will use the extension from the original file. For example, the following two calls to latexindent.pl are equivalent:

```
cmh:~$ latexindent.pl myfile.tex -o=output

cmh:~$ latexindent.pl myfile.tex -o=output.tex
```

You can call the -o switch using a + symbol at the beginning; this will concatenate the name of the input file and the text given to the -o switch. For example, the following two calls to latexindent.pl are equivalent:

```
cmh:~$ latexindent.pl myfile.tex -o=new

cmh:~$ latexindent.pl myfile.tex -o=myfilenew.tex
```

You can call the -o switch using a ++ symbol at the end of the name of your output file; this tells latexindent.pl to search successively for the name of your output file concatenated with 0, 1,… while the name of the output file exists. For example,

```
cmh:~$ latexindent.pl myfile.tex -o=output++
```

tells latexindent.pl to output to output0.tex, but if it exists then output to output1.tex, and so on.

Calling latexindent.pl with simply

```
cmh:~$ latexindent.pl myfile.tex -o=++
```

Users of version 2.* should note the subtle change in syntax
tells it to output to myfile0.tex, but if it exists then output to myfile1.tex and so on.

The + and ++ feature of the -o switch can be combined; for example, calling

```
cmh:~$ latexindent.pl myfile.tex -o=+out++
```

tells latexindent.pl to output to myfileout0.tex, but if it exists, then try myfileout1.tex, and so on.

There is no need to specify a file extension when using the ++ feature, but if you wish to, then you should include it after the ++ symbols, for example

```
cmh:~$ latexindent.pl myfile.tex -o=+out++.tex
```

See appendix H on page 148 for details of how the interface has changed from Version 2.2 to Version 3.0 for this flag.

-s, --silent

```
cmh:~$ latexindent.pl -s myfile.tex
```

Silent mode: no output will be given to the terminal.

-t, --trace

```
cmh:~$ latexindent.pl -t myfile.tex
```

Tracing mode: verbose output will be given to indent.log. This is useful if latexindent.pl has made a mistake and you’re trying to find out where and why. You might also be interested in learning about latexindent.pl’s thought process – if so, this switch is for you, although it should be noted that, especially for large files, this does affect performance of the script.

-tt, --ttrace

```
cmh:~$ latexindent.pl -tt myfile.tex
```

More detailed tracing mode: this option gives more details to indent.log than the standard trace option (note that, even more so than with -t, especially for large files, performance of the script will be affected).

-l, --local[=myyaml.yaml,other.yaml, ...]

```
cmh:~$ latexindent.pl -l myfile.tex
```

latexindent.pl will always load defaultSettings.yaml (rhymes with camel) and if it is called with the -l switch and it finds localSettings.yaml in the same directory as myfile.tex, then, if not found, it looks for localSettings.yaml (and friends, see Section 4.2 on page 17) in the current
3.1 From the command line

working directory, then these settings will be added to the indentation scheme. Information will be
given in indent.log on the success or failure of loading localSettings.yaml.

The -l flag can take an optional parameter which details the name (or names separated by com-
mas) of a YAML file(s) that resides in the same directory as myfile.tex; you can use this op-
tion if you would like to load a settings file in the current working directory that is not called
localSettings.yaml. In fact, you can specify both relative and absolute paths for your YAML files;
for example

```bash
$ latexindent.pl -l=./myyaml.yaml myfile.tex
$ latexindent.pl -l=/home/cmhuies/Desktop/myyaml.yaml myfile.tex
$ latexindent.pl -l=C:\Users\cmhuies\Desktop\myyaml.yaml myfile.tex
```

You will find a lot of other explicit demonstrations of how to use the -l switch throughout this
documentation,

You can call the -l switch with a ‘+’ symbol either before or after another YAML file; for example:

```bash
$ latexindent.pl -l=+myyaml.yaml myfile.tex
$ latexindent.pl -l "+myyaml.yaml" myfile.tex
```

which translate, respectively, to

```bash
$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
$ latexindent.pl -l=myyaml.yaml,localSettings.yaml myfile.tex
```

Note that the following is not allowed:

```bash
$ latexindent.pl -l+myyaml.yaml myfile.tex
```

and

```bash
$ latexindent.pl -l + myyaml.yaml myfile.tex
```

will only load localSettings.yaml, and myyaml.yaml will be ignored. If you wish to use spaces
between any of the YAML settings, then you must wrap the entire list of YAML files in quotes, as
demonstrated above.

You may also choose to omit the yaml extension, such as

```bash
$ latexindent.pl -l=localSettings,myyaml myfile.tex
```

-y, --yaml=settings

```bash
$ latexindent.pl myfile.tex -y="defaultIndent:␣'␣'"
$ latexindent.pl myfile.tex -y="defaultIndent:␣'␣',maximumIndentation:'␣'"
$ latexindent.pl myfile.tex -y="indentRules:␣one:␣'				'"
$ latexindent.pl myfile.tex
  -y='modifyLineBreaks:environments:EndStartsOnOwnLine:3' -m
$ latexindent.pl myfile.tex
  -y='modifyLineBreaks:environments:one:EndStartsOnOwnLine:3' -m
```
3.1 From the command line

You can specify YAML settings from the command line using the \(-y\) or \(--yaml\) switch; sample demonstrations are given above. Note, in particular, that multiple settings can be specified by separating them via commas. There is a further option to use a \;\) to separate fields, which is demonstrated in Section 4.3 on page 18.

Any settings specified via this switch will be loaded after any specified using the \(-l\) switch. This is discussed further in Section 4.4 on page 19.

\(-d\), \(--onlydefault\)

```bash
   cmh:~$ latexindent.pl -d myfile.tex
```

Only defaultSettings.yaml: you might like to read Section 5 before using this switch. By default, latexindent.pl will always search for indentconfig.yaml or .indentconfig.yaml in your home directory. If you would prefer it not to do so then (instead of deleting or renaming indentconfig.yaml or .indentconfig.yaml) you can simply call the script with the \(-d\) switch; note that this will also tell the script to ignore localSettings.yaml even if it has been called with the \(-l\) switch; latexindent.pl will also ignore any settings specified from the \(-y\) switch.

\(-c\), \(--cruft=<directory>\)

```bash
   cmh:~$ latexindent.pl -c=/path/to/directory/ myfile.tex
```

If you wish to have backup files and indent.log written to a directory other than the current working directory, then you can send these ‘cruft’ files to another directory. Note the use of a trailing forward slash.

\(-g\), \(--logfile=<name of log file>\)

```bash
   cmh:~$ latexindent.pl -g=other.log myfile.tex
   cmh:~$ latexindent.pl --logfile other.log myfile.tex
```

By default, latexindent.pl reports information to indent.log, but if you wish to change the name of this file, simply call the script with your chosen name after the \(-g\) switch as demonstrated above.

If latexindent.pl can not open the log file that you specify, then the script will operate, and no log file will be produced; this might be helpful to users who wish to specify the following, for example

```bash
   cmh:~$ latexindent.pl -g /dev/null myfile.tex
```

\(-sl\), \(--screenlog\)

```bash
   cmh:~$ latexindent.pl -sl myfile.tex
   cmh:~$ latexindent.pl --screenlog myfile.tex
```

Using this option tells latexindent.pl to output the log file to the screen, as well as to your chosen log file.

\(-m\), \(--modifylinebreaks\)

```bash
   cmh:~$ latexindent.pl -m myfile.tex
   cmh:~$ latexindent.pl --modifylinebreaks myfile.tex
```
One of the most exciting developments in Version 3.0 is the ability to modify line breaks; for full details see Section 6 on page 68.

\texttt{latexindent.pl} can also be called on a file without the file extension, for example

\begin{verbatim}
$ latexindent.pl myfile
\end{verbatim}

and in which case, you can specify the order in which extensions are searched for; see Listing 16 on page 20 for full details.

\textbf{STDIN}

\begin{verbatim}
$ cat myfile.tex | latexindent.pl
$ cat myfile.tex | latexindent.pl -
\end{verbatim}

\texttt{latexindent.pl} will allow input from STDIN, which means that you can pipe output from other commands directly into the script. For example assuming that you have content in \texttt{myfile.tex}, then the above command will output the results of operating upon \texttt{myfile.tex}.

If you wish to use this feature with your own local settings, via the \texttt{-l} switch, then you should finish your call to \texttt{latexindent.pl} with a \texttt{-} sign:

\begin{verbatim}
$ cat myfile.tex | latexindent.pl -l=mysettings.yaml -
\end{verbatim}

Similarly, if you simply type \texttt{latexindent.pl} at the command line, then it will expect (STDIN) input from the command line.

\begin{verbatim}
$ latexindent.pl
\end{verbatim}

Once you have finished typing your input, you can press

- \texttt{CTRL+D} on Linux
- \texttt{CTRL+Z} followed by \texttt{ENTER} on Windows

to signify that your input has finished. Thanks to [5] for an update to this feature.

\texttt{-r, --replacement}

\begin{verbatim}
$ latexindent.pl -r myfile.tex
$ latexindent.pl --replacement myfile.tex
\end{verbatim}

You can call \texttt{latexindent.pl} with the \texttt{-r} switch to instruct it to perform replacements/substitutions on your file; full details and examples are given in Section 7 on page 113.

\texttt{-rv, --replacementrespectverb}

\begin{verbatim}
$ latexindent.pl -rv myfile.tex
$ latexindent.pl --replacementrespectverb myfile.tex
\end{verbatim}

You can instruct \texttt{latexindent.pl} to perform replacements/substitutions by using the \texttt{-rv} switch, but will respect \textit{verbatim} code blocks; full details and examples are given in Section 7 on page 113.

\texttt{-rr, --onlyreplacement}
You can instruct `latexindent.pl` to skip all of its other indentation operations and only perform replacements/substitutions by using the `-rr` switch; full details and examples are given in Section 7 on page 113.

- `-k`, `-check`

You can instruct `latexindent.pl` to check if the text after indentation matches that given in the original file.

The exit code of `latexindent.pl` is 0 by default. If you use the `-k` switch then

- if the text after indentation matches that given in the original file, then the exit code is 0;
- if the text after indentation does not match that given in the original file, then the exit code is 1.

The value of the exit code may be important to those wishing to, for example, check the status of the indentation in continuous integration tools such as GitHub Actions. Full details of the exit codes of `latexindent.pl` are given in Table 1.

A simple `diff` will be given in `indent.log`.

- `-kv`, `-checkv`

The check verbose switch is exactly the same as the `-k` switch, except that it is verbose, and it will output the (simple) diff to the terminal, as well as to `indent.log`.

- `-n`, `-lines=MIN-MAX`

The `lines` switch instructs `latexindent.pl` to operate only on specific line ranges within `myfile.tex`.

Complete demonstrations are given in Section 8.

3.2 From arara

Using `latexindent.pl` from the command line is fine for some folks, but others may find it easier to use from arara; you can find the arara rule for `latexindent.pl` and its associated documentation at [4].

3.3 Summary of exit codes

Assuming that you call `latexindent.pl` on `myfile.tex`

```
cmh:~$ latexindent.pl myfile.tex
```

then `latexindent.pl` can exit with the exit codes given in Table 1.
### 3.3 Summary of exit codes

<table>
<thead>
<tr>
<th>exit code</th>
<th>indentation</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>✔️</td>
<td>success; if <code>-k</code> or <code>-kv</code> active, indented text matches original</td>
</tr>
<tr>
<td>0</td>
<td>✗</td>
<td>success; if <code>-version</code> or <code>-help</code>, no indentation performed</td>
</tr>
<tr>
<td>1</td>
<td>✔️</td>
<td>success, and <code>-k</code> or <code>-kv</code> active; indented text different from original</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>failure, <code>defaultSettings.yaml</code> could not be read</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>failure, <code>myfile.tex</code> not found</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>failure, <code>myfile.tex</code> exists but cannot be read</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>failure, <code>-w</code> active, and back-up file cannot be written</td>
</tr>
<tr>
<td>6</td>
<td>✗</td>
<td>failure, <code>-c</code> active, and cruft directory does not exist</td>
</tr>
</tbody>
</table>

**Table 1: Exit codes for `latexindent.pl`**
SECTION 4

indentconfig.yaml, local settings and the -y switch

The behaviour of latexindent.pl is controlled from the settings specified in any of the YAML files that you tell it to load. By default, latexindent.pl will only load defaultSettings.yaml, but there are a few ways that you can tell it to load your own settings files.

4.1 indentconfig.yaml and .indentconfig.yaml

latexindent.pl will always check your home directory for indentconfig.yaml and .indentconfig.yaml (unless it is called with the -d switch), which is a plain text file you can create that contains the absolute paths for any settings files that you wish latexindent.pl to load. There is no difference between indentconfig.yaml and .indentconfig.yaml, other than the fact that .indentconfig.yaml is a 'hidden' file; thank you to [11] for providing this feature. In what follows, we will use indentconfig.yaml, but it is understood that this could equally represent .indentconfig.yaml. If you have both files in existence then indentconfig.yaml takes priority.

For Mac and Linux users, their home directory is /username while Windows (Vista onwards) is C:\Users\username. Listing 12 shows a sample indentconfig.yaml file.

```
# Paths to user settings for latexindent.pl
#
# Note that the settings will be read in the order you
# specify here- each successive settings file will overwrite
# the variables that you specify

paths:
- /home/cmhughes/Documents/yamlfiles/mysettings.yaml
- /home/cmhughes/folder/othersettings.yaml
- /some/other/folder/anynameyouwant.yaml
- C:\Users\chughes\Documents\mysettings.yaml
- C:\Users\chughes\Desktop\test spaces\more spaces.yaml
```

Note that the .yaml files you specify in indentconfig.yaml will be loaded in the order in which you write them. Each file doesn't have to have every switch from defaultSettings.yaml; in fact, I recommend that you only keep the switches that you want to change in these settings files.

To get started with your own settings file, you might like to save a copy of defaultSettings.yaml in another directory and call it, for example, mysettings.yaml. Once you have added the path to indentconfig.yaml you can change the switches and add more code-block names to it as you see fit – have a look at Listing 13 for an example that uses four tabs for the default indent, adds the tabbing environment/command to the list of environments that contains alignment delimiters; you might also like to refer to the many YAML files detailed throughout the rest of this documentation.

---

2If you're not sure where to put indentconfig.yaml, don't worry latexindent.pl will tell you in the log file exactly where to put it assuming it doesn't exist already.
LISTING 13: mysettings.yaml (example)

```yaml
# Default value of indentation
defaultIndent: "\t\t\t\t"

# environments that have tab delimiters, add more
# as needed
lookForAlignDelims:
  tabbing: 1
```

You can make sure that your settings are loaded by checking indent.log for details – if you have specified a path that latexindent.pl doesn’t recognise then you’ll get a warning, otherwise you’ll get confirmation that latexindent.pl has read your settings file.

**Warning!**

When editing `.yaml` files it is extremely important to remember how sensitive they are to spaces. I highly recommend copying and pasting from defaultSettings.yaml when you create your first `whatevernameyoulike.yaml` file.

If latexindent.pl cannot read your `.yaml` file it will tell you so in indent.log.

If you find that latexindent.pl does not read your YAML file, then it might be as a result of the default commandline encoding not being UTF-8; normally this will only occur for Windows users. In this case, you might like to explore the encoding option for indentconfig.yaml as demonstrated in Listing 14.

LISTING 14: The encoding option for indentconfig.yaml

```yaml
encoding: GB2312
paths:
  - D:\cmh\latexindent.yaml
```

Thank you to [25] for this contribution; please see appendix F on page 146 and details within [24] for further information.

### 4.2 localSettings.yaml and friends

The `-l` switch tells latexindent.pl to look for `localSettings.yaml` and/or friends in the same directory as `myfile.tex`. For example, if you use the following command

```bash
$ latexindent.pl -l myfile.tex
```

then latexindent.pl will search for and then, assuming they exist, load each of the following files in the following order:

1. localSettings.yaml
2. latexindent.yaml
3. .localSettings.yaml
4. .latexindent.yaml

These files will be assumed to be in the same directory as `myfile.tex`, or otherwise in the current working directory. You do not need to have all of the above files, usually just one will be sufficient. In what follows, whenever we refer to `localSettings.yaml` it is assumed that it can mean any of the four named options listed above.

---

3 Windows users may find that they have to end `.yaml` files with a blank line.
If you’d prefer to name your localSettings.yaml file something different, (say, mysettings.yaml as in Listing 13) then you can call latexindent.pl using, for example,

```
cmh:~$ latexindent.pl -l=mysettings.yaml myfile.tex
```

Any settings file(s) specified using the -l switch will be read after defaultSettings.yaml and, assuming they exist, any user setting files specified in indentconfig.yaml.

Your settings file can contain any switches that you’d like to change; a sample is shown in Listing 15, and you’ll find plenty of further examples throughout this manual.

**LISTING 15: localSettings.yaml (example)**

```yaml
# verbatim environments - environments specified
# here will not be changed at all!
verbatimEnvironments:
  cmhenvironment: 0
  myenv: 1
```

You can make sure that your settings file has been loaded by checking indent.log for details; if it can not be read then you receive a warning, otherwise you’ll get confirmation that latexindent.pl has read your settings file.

### 4.3 The -y|yaml switch

You may use the -y switch to load your settings; for example, if you wished to specify the settings from Listing 15 using the -y switch, then you could use the following command:

```
cmh:~$ latexindent.pl -y="verbatimEnvironments:cmhenvironment:0;myenv:1" myfile.tex
```

Note the use of ; to specify another field within verbatimEnvironments. This is shorthand, and equivalent, to using the following command:

```
cmh:~$ latexindent.pl -y="verbatimEnvironments:cmhenvironment:0,verbatimEnvironments:myenv:1" myfile.tex
```

You may, of course, specify settings using the -y switch as well as, for example, settings loaded using the -l switch; for example,

```
cmh:~$ latexindent.pl -l=mysettings.yaml -y="verbatimEnvironments:cmhenvironment:0;myenv:1" myfile.tex
```

Any settings specified using the -y switch will be loaded after any specified using indentconfig.yaml and the -l switch.

If you wish to specify any regex-based settings using the -y switch, it is important not to use quotes surrounding the regex; for example, with reference to the 'one sentence per line' feature (Section 6.5 on page 88) and the listings within Listing 345 on page 89, the following settings give the option to have sentences end with a semicolon

```
cmh:~$ latexindent.pl -m
--yaml='modifyLineBreaks:oneSentencePerLine:sentencesEndWith:other:',
```

4.4 Settings load order

latexindent.pl loads the settings files in the following order:

1. defaultSettings.yaml is always loaded, and can not be renamed;
2. anyUserSettings.yaml and any other arbitrarily-named files specified in indentconfig.yaml;
3. localSettings.yaml but only if found in the same directory as myfile.tex and called with -l switch; this file can be renamed, provided that the call to latexindent.pl is adjusted accordingly (see Section 4.2). You may specify both relative and absolute paths to other YAML files using the -l switch, separating multiple files using commas;
4. any settings specified in the -y switch.

A visual representation of this is given in Figure 1.

---

**Figure 1:** Schematic of the load order described in Section 4.4; solid lines represent mandatory files, dotted lines represent optional files. indentconfig.yaml can contain as many files as you like. The files will be loaded in order; if you specify settings for the same field in more than one file, the most recent takes priority.
SECTION 5

defaultSettings.yaml

latexindent.pl loads its settings from defaultSettings.yaml. The idea is to separate the behaviour of the script from the internal working – this is very similar to the way that we separate content from form when writing our documents in \LaTeX.

If you look in defaultSettings.yaml you’ll find the switches that govern the behaviour of latexindent.pl. If you’re not sure where defaultSettings.yaml resides on your computer, don’t worry as indent.log will tell you where to find it. defaultSettings.yaml is commented, but here is a description of what each switch is designed to do. The default value is given in each case; whenever you see integer in this section, assume that it must be greater than or equal to 0 unless otherwise stated.

For most of the settings in defaultSettings.yaml that are specified as integers, then we understand 0 to represent ‘off’ and 1 to represent ‘on’. For fields that allow values other than 0 or 1, it is hoped that the specific context and associated commentary should make it clear which values are allowed.

### fileExtensionPreference: (fields)

latexindent.pl can be called to act on a file without specifying the file extension. For example we can call

```
cmh:~$ latexindent.pl myfile
```

in which case the script will look for myfile with the extensions specified in fileExtensionPreference in their numeric order. If no match is found, the script will exit. As with all of the fields, you should change and/or add to this as necessary.

<table>
<thead>
<tr>
<th>Listing 16: fileExtensionPreference</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 fileExtensionPreference:</td>
</tr>
<tr>
<td>45 .tex: 1</td>
</tr>
<tr>
<td>46 .sty: 2</td>
</tr>
<tr>
<td>47 .cls: 3</td>
</tr>
<tr>
<td>48 .bib: 4</td>
</tr>
</tbody>
</table>

Calling latexindent.pl myfile with the (default) settings specified in Listing 16 means that the script will first look for myfile.tex, then myfile.sty, myfile.cls, and finally myfile.bib in order.

### 5.1 Backup and log file preferences

If you call latexindent.pl with the `-w` switch (to overwrite myfile.tex) then it will create a backup file before doing any indentation; the default extension is .bak, so, for example, myfile.bak0 would be created when calling latexindent.pl myfile.tex for the first time.

By default, every time you subsequently call latexindent.pl with the `-w` to act upon myfile.tex, it will create successive back up files: myfile.bak1, myfile.bak2, etc.

---

Throughout this manual, listings shown with line numbers represent code taken directly from defaultSettings.yaml.
5.1 Backup and log file preferences

onlyOneBackUp: (integer)

If you don’t want a backup for every time that you call \texttt{latexindent.pl} (so you don’t want \texttt{myfile.bak1}, \texttt{myfile.bak2}, etc) and you simply want \texttt{myfile.bak} (or whatever you chose backupExtension to be) then change \texttt{onlyOneBackUp} to 1; the default value of \texttt{onlyOneBackUp} is 0.

maxNumberOfBackUps: (integer)

Some users may only want a finite number of backup files, say at most 3, in which case, they can change this switch. The smallest value of \texttt{maxNumberOfBackUps} is 0 which will not prevent backup files being made; in this case, the behaviour will be dictated entirely by \texttt{onlyOneBackUp}. The default value of \texttt{maxNumberOfBackUps} is 0.

cycleThroughBackUps: (integer)

Some users may wish to cycle through backup files, by deleting the oldest backup file and keeping only the most recent; for example, with \texttt{maxNumberOfBackUps: 4}, and \texttt{cycleThroughBackUps} set to 1 then the \texttt{copy} procedure given below would be obeyed.

\begin{verbatim}
cmh:~$ copy myfile.bak1 to myfile.bak0
cmh:~$ copy myfile.bak2 to myfile.bak1
cmh:~$ copy myfile.bak3 to myfile.bak2
cmh:~$ copy myfile.bak4 to myfile.bak3
\end{verbatim}

The default value of \texttt{cycleThroughBackUps} is 0.

logFilePreferences: (fields)

\texttt{latexindent.pl} writes information to \texttt{indent.log}, some of which can be customized by changing \texttt{logFilePreferences}; see Listing 17. If you load your own user settings (see Section 4 on page 16) then \texttt{latexindent.pl} will detail them in \texttt{indent.log}; you can choose not to have the details logged by switching \texttt{showEveryYamlRead} to 0. Once all of your settings have been loaded, you can see the amalgamated settings in the log file by switching \texttt{showAmalgamatedSettings} to 1, if you wish.

\begin{verbatim}
88 logFilePreferences:
89    showEveryYamlRead: 1
90    showAmalgamatedSettings: 0
91    showDecorationStartCodeBlockTrace: 0
92    showDecorationFinishCodeBlockTrace: 0
93    endLogFileWith: '--------------'
94    showGitHubInfoFooter: 1
95    Dumper:
96      Terse: 1
97      Indent: 1
98      Useqq: 1
99      Deparse: 1
100     Quotekeys: 0
101     Sortkeys: 1
102     Pair: " => "
\end{verbatim}

When either of the trace modes (see page 10) are active, you will receive detailed information in \texttt{indent.log}. You can specify character strings to appear before and after the notification of a found code block using, respectively, \texttt{showDecorationStartCodeBlockTrace} and \texttt{showDecorationFinishCodeBlockTrace}. A demonstration is given in appendix E on page 145.
The log file will end with the characters given in endLogFileWith, and will report the GitHub address of latexindent.pl to the log file if showGitHubInfoFooter is set to 1.

Note: latexindent.pl no longer uses the log4perl module to handle the creation of the log file.

Some of the options for Perl's Dumper module can be specified in Listing 17; see [9] and [8] for more information. These options will mostly be helpful for those calling latexindent.pl with the -tt option described in Section 3.1.

5.2 Verbatim code blocks

A field that contains a list of environments that you would like left completely alone – no indentation will be performed on environments that you have specified in this field, see Listing 18.

<table>
<thead>
<tr>
<th>Listing 18: verbatimEnvironments</th>
</tr>
</thead>
<tbody>
<tr>
<td>106 verbatimEnvironments:</td>
</tr>
<tr>
<td>107      verbatim: 1</td>
</tr>
<tr>
<td>108     lstlisting: 1</td>
</tr>
<tr>
<td>109       minted: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 19: verbatimCommands</th>
</tr>
</thead>
<tbody>
<tr>
<td>112 verbatimCommands:</td>
</tr>
<tr>
<td>113      verb: 1</td>
</tr>
<tr>
<td>114     lstinline: 1</td>
</tr>
</tbody>
</table>

Note that if you put an environment in verbatimEnvironments and in other fields such as lookForAlignDelims or noAdditionalIndent then latexindent.pl will always prioritize verbatimEnvironments.

You can, optionally, specify the verbatim field using the name field which takes a regular expression as its argument; thank you to [35] for contributing this feature.

For demonstration, then assuming that your file contains the environments latexcode, latexcode*, pythoncode and pythoncode*, then the listings given in Listings 20 and 21 are equivalent.

<table>
<thead>
<tr>
<th>Listing 20: nameAsRegex1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbatimEnvironments:</td>
</tr>
<tr>
<td>latexcode: 1</td>
</tr>
<tr>
<td>latexcode*: 1</td>
</tr>
<tr>
<td>pythoncode: 1</td>
</tr>
<tr>
<td>pythoncode*: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 21: nameAsRegex2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbatimEnvironments:</td>
</tr>
<tr>
<td>nameAsRegex:</td>
</tr>
<tr>
<td>name: '\w+code*?'</td>
</tr>
<tr>
<td>lookForThis: 1</td>
</tr>
</tbody>
</table>

With reference to Listing 21:

- the name field as specified here means any word followed by the word code, optionally followed by *;
- we have used nameAsRegex to identify this field, but you can use any description you like;
- the lookForThis field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

A field that contains a list of commands that are verbatim commands, for example \lstinline; any commands populated in this field are protected from line breaking routines (only relevant if the -m is active, see Section 6 on page 68).

With reference to Listing 19, by default latexindent.pl looks for \verb immediately followed by another character, and then it takes the body as anything up to the next occurrence of the character; this means that, for example, \verb!x+3! is treated as a verbatimCommands.

You can, optionally, specify the verbatimCommands field using the name field which takes a regular expression as its argument; thank you to [35] for contributing this feature.

For demonstration, then assuming that your file contains the commands verbinline, myinline then the listings given in Listings 22 and 23 are equivalent.
5.2 Verbatim code blocks

With reference to Listing 23:

- the *name* field as specified here means *any word followed by the word inline*;
- we have used *nameAsRegex* to identify this field, but you can use any description you like;
- the *lookForThis* field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

If you have a block of code that you don't want *latexindent.pl* to touch (even if it is *not* a verbatim-like environment) then you can wrap it in an environment from *noIndentBlock*; you can use any name you like for this, provided you populate it as demonstrate in Listing 24.

Of course, you don't want to have to specify these as null environments in your code, so you use them with a comment symbol, `%`, followed by as many spaces (possibly none) as you like; see Listing 25 for example.

Important note: it is assumed that the *noindent* block statements specified in this way appear on their own line.

The *noIndentBlock* fields can also be specified in terms of *begin* and *end* fields. We use the code in Listing 26 to demonstrate this feature.

The settings given in Listings 27 and 28 are equivalent:
Upon running the commands

cmh:~$ latexindent.pl -l noindent1.yaml noindent1

then we receive the output given in Listing 30.

Upon running the commands

cmh:~$ latexindent.pl -l noindent2.yaml noindent1

cmh:~$ latexindent.pl -l noindent3.yaml noindent1

then we receive the output given in Listing 30.

LISTING 30: noIndentBlock1.tex using Listing 27 or Listing 28

some before text
  this code
  won’t
  be touched
  by
  latexindent.pl!
some after text

The begin, body and end fields for noIndentBlock are all regular expressions. If the body field is not specified, then it takes a default value of .*? which is written explicitly in Listing 27. In this context, we interpret .*? in words as the fewest number of characters (possibly none) until the ‘end’ field is reached.

The lookForThis field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

Using Listing 29 demonstrates setting lookForThis to 0 (off); running the command

cmh:~$ latexindent.pl -l noindent3.yaml noindent1

gives the output in Listing 31.

LISTING 31: noIndentBlock1.tex using Listing 29

some before text
  this code
  won’t
  be touched
  by
  latexindent.pl!
some after text

We will demonstrate this feature later in the documentation in Listing 537.

You can, optionally, specify the noIndentBlock field using the name field which takes a regular expression as its argument; thank you to [35] for contributing this feature.

For demonstration, then assuming that your file contains the environments testnoindent, testnoindent* then the listings given in Listings 32 and 33 are equivalent.
With reference to Listing 33:

- the `name` field as specified here means *any word followed by the word noindent, optionally followed by *;
- we have used `nameAsRegex` to identify this field, but you can use any description you like;
- the `lookForThis` field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

### 5.3 filecontents and preamble

**fileContentsEnvironments**: `{field}`

Before `latexindent.pl` determines the difference between preamble (if any) and the main document, it first searches for any of the environments specified in `fileContentsEnvironments`, see Listing 34. The behaviour of `latexindent.pl` on these environments is determined by their location (preamble or not), and the value `indentPreamble`, discussed next.

#### Listing 34: fileContentsEnvironments

```yaml
fileContentsEnvironments:
  filecontents: 1
  filecontents*: 1
```

**indentPreamble**: `0|1`

The preamble of a document can sometimes contain some trickier code for `latexindent.pl` to operate upon. By default, `latexindent.pl` won't try to operate on the preamble (as `indentPreamble` is set to 0, by default), but if you'd like `latexindent.pl` to try then change `indentPreamble` to 1.

**lookForPreamble**: `{fields}`

Not all files contain preamble; for example, sty, cls and bib files typically do not. Referencing Listing 35, if you set, for example, `.tex` to 0, then regardless of the setting of the value of `indentPreamble`, preamble will not be assumed when operating upon `.tex` files.

#### Listing 35: lookForPreamble

```yaml
lookForPreamble:
  .tex: 1
  .sty: 0
  .cls: 0
  .bib: 0
```

**preambleCommandsBeforeEnvironments**: `0|1`

Assuming that `latexindent.pl` is asked to operate upon the preamble of a document, when this switch is set to 0 then environment code blocks will be sought first, and then command code blocks. When this switch is set to 1, commands will be sought first. The example that first motivated this switch contained the code given in Listing 36.
5.4 Indentation and horizontal space

**defaultIndent**: (horizontal space)

This is the default indentation used in the absence of other details for the code block with which we are working. The default value is \t which means a tab; we will explore customisation beyond defaultIndent in Section 5.8 on page 46.

If you're interested in experimenting with latexindent.pl then you can remove all indentation by setting defaultIndent: "".

**removeTrailingWhitespace**: (fields)

Trailing white space can be removed both before and after processing the document, as detailed in Listing 37; each of the fields can take the values 0 or 1. See Listings 428 to 430 on page 102 for before and after results. Thanks to [31] for providing this feature.

<table>
<thead>
<tr>
<th>Listing 37: removeTrailingWhitespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>removeTrailingWhitespace: 1</td>
</tr>
<tr>
<td>beforeProcessing: 0</td>
</tr>
<tr>
<td>afterProcessing: 1</td>
</tr>
</tbody>
</table>

N: 2017-06-28

You can specify removeTrailingWhitespace simply as 0 or 1, if you wish; in this case, latexindent.pl will set both beforeProcessing and afterProcessing to the value you specify; see Listing 38.

5.5 Aligning at delimiters

**lookForAlignDelims**: (fields)

This contains a list of code blocks that are operated upon in a special way by latexindent.pl (see Listing 39). In fact, the fields in lookForAlignDelims can actually take two different forms: the basic version is shown in Listing 39 and the advanced version in Listing 42; we will discuss each in turn.

<table>
<thead>
<tr>
<th>Listing 39: lookForAlignDelims (basic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>tabular: 1</td>
</tr>
<tr>
<td>tabularx: 1</td>
</tr>
<tr>
<td>longtable: 1</td>
</tr>
<tr>
<td>array: 1</td>
</tr>
<tr>
<td>matrix: 1</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Specifying code blocks in this field instructs latexindent.pl to try and align each column by its alignment delimiters. It does have some limitations (discussed further in Section 10), but in many cases it will produce results such as those in Listings 40 and 41.
If you find that \texttt{latexindent.pl} does not perform satisfactorily on such environments then you can set the relevant key to 0, for example tabular: 0; alternatively, if you just want to ignore specific instances of the environment, you could wrap them in something from \texttt{noIndentBlock} (see Listing 24 on page 23).

If, for example, you wish to remove the alignment of the | within a delimiter-aligned block, then the advanced form of \texttt{lookForAlignDelims} shown in Listing 42 is for you.

Note that you can use a mixture of the basic and advanced form: in Listing 42 \texttt{tabular} and \texttt{tabularx} are advanced and \texttt{longtable} is basic. When using the advanced form, each field should receive at least 1 sub-field, and \texttt{can} (but does not have to) receive any of the following fields:

- \texttt{delims}: binary switch (0 or 1) equivalent to simply specifying, for example, \texttt{tabular: 1} in the basic version shown in Listing 39. If \texttt{delims} is set to 0 then the align at ampersand routine will not be called for this code block (default: 1);
- \texttt{alignDoubleBackSlash}: binary switch (0 or 1) to determine if | should be aligned (default: 1);
- \texttt{spacesBeforeDoubleBackSlash}: optionally, specifies the number (integer \(\geq 0\)) of spaces to be inserted before | (default: 1);
- \texttt{multiColumnGrouping}: binary switch (0 or 1) that details if \texttt{latexindent.pl} should group columns above and below a \texttt{\multicolumn} command (default: 0);
- \texttt{alignRowsWithoutMaxDelims}: binary switch (0 or 1) that details if rows that do not contain the maximum number of delimeters should be formatted so as to have the ampersands aligned (default: 1);
- \texttt{spacesBeforeAmpersand}: optionally specifies the number (integer \(\geq 0\)) of spaces to be placed before ampersands (default: 1);
- \texttt{spacesAfterAmpersand}: optionally specifies the number (integer \(\geq 0\)) of spaces to be placed after ampersands (default: 1);
- \texttt{justification}: optionally specifies the justification of each cell as either \texttt{left} or \texttt{right} (default: \texttt{left});
• alignFinalDoubleBackSlash optionally specifies if the final double back slash should be used for alignment (default: 0);

• dontMeasure optionally specifies if user-specified cells, rows or the largest entries should not be measured (default: 0);

• delimiterRegEx optionally specifies the pattern matching to be used for the alignment delimiter (default: '(?<!\)(&)');

• delimiterJustification optionally specifies the justification for the alignment delimiters (default: left); note that this feature is only useful if you have delimiters of different lengths in the same column, discussed in Section 5.5.4;

• lookForChildCodeBlocks optionally instructs latexindent.pl to search for child code blocks or not (default: 1), discussed in Section 5.5.5.

We will explore most of these features using the file tabular2.tex in Listing 43 (which contains a \multicolumn command), and the YAML files in Listings 44 to 50; we will explore alignFinalDoubleBackSlash in Listing 71; the dontMeasure feature will be described in Section 5.5.3, and delimiterRegEx in Section 5.5.4.

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & six & \\
seven & \\
\end{tabular}

On running the commands

\begin{verbatim}
\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & six & \\
seven & \\
\end{tabular}
\end{verbatim}
we obtain the respective outputs given in Listings 51 to 58.

**Listing 51: tabular2.tex default output**

```latex
\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}
```

**Listing 52: tabular2.tex using Listing 44**

```latex
\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}
```

**Listing 53: tabular2.tex using Listing 45**

```latex
\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}
```

**Listing 54: tabular2.tex using Listings 44 and 46**

```latex
\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}
```
5.5 Aligning at delimiters

<table>
<thead>
<tr>
<th>Listing 55: tabular2.tex using Listings 44 and 47</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cccc} \hline A &amp; B &amp; C &amp; D \ AAA &amp; BBB &amp; CCC &amp; DDD \ \multicolumn{2}{c}{first heading} &amp; \multicolumn{2}{c}{second heading} \ one &amp; two &amp; three &amp; four \ five &amp; six &amp; &amp; \ seven &amp; &amp; six &amp; \ \end{tabular}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 56: tabular2.tex using Listings 44 and 48</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cccc} \hline A &amp; B &amp; C &amp; D \ AAA &amp; BBB &amp; CCC &amp; DDD \ \multicolumn{2}{c}{first heading} &amp; \multicolumn{2}{c}{second heading} \ one &amp; two &amp; three &amp; four \ five &amp; &amp; six &amp; \ seven &amp; &amp; \ \end{tabular}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 57: tabular2.tex using Listings 44 and 49</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cccc} \hline A &amp; B &amp; C &amp; D \ AAA &amp; BBB &amp; CCC &amp; DDD \ \multicolumn{2}{c}{first heading} &amp; \multicolumn{2}{c}{second heading} \ one &amp; two &amp; three &amp; four \ five &amp; &amp; six &amp; \ seven &amp; &amp; \ \end{tabular}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 58: tabular2.tex using Listings 44 and 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cccc} \hline A &amp; B &amp; C &amp; D \ AAA &amp; BBB &amp; CCC &amp; DDD \ \multicolumn{2}{c}{first heading} &amp; \multicolumn{2}{c}{second heading} \ one &amp; two &amp; three &amp; four \ five &amp; &amp; six &amp; \ seven &amp; &amp; \ \end{tabular}</td>
</tr>
</tbody>
</table>

Notice in particular:

- in both Listings 51 and 52 all rows have been aligned at the ampersand, even those that do not contain the maximum number of ampersands (3 ampersands, in this case);
- in Listing 51 the columns have been aligned at the ampersand;
- in Listing 52 the \multicolumn command has grouped the 2 columns beneath and above it, because multiColumnGrouping is set to 1 in Listing 44;
- in Listing 53 rows 3 and 6 have not been aligned at the ampersand, because alignRowsWithoutMaxDelims has been set to 0 in Listing 45; however, the \ have still been aligned;
- in Listing 54 the columns beneath and above the \multicolumn commands have been grouped (because multiColumnGrouping is set to 1), and there are at least 4 spaces before each aligned ampersand because spacesBeforeAmpersand is set to 4;
- in Listing 55 the columns beneath and above the \multicolumn commands have been grouped (because multiColumnGrouping is set to 1), and there are at least 4 spaces after each aligned ampersand because spacesAfterAmpersand is set to 4;
• in Listing 56 the `\` have not been aligned, because `alignDoubleBackSlash` is set to 0, otherwise the output is the same as Listing 52;

• in Listing 57 the `\` have been aligned, and because `spacesBeforeDoubleBackSlash` is set to 0, there are no spaces ahead of them; the output is otherwise the same as Listing 52;

• in Listing 58 the cells have been right-justified; note that cells above and below the `\multicol` statements have still been group correctly, because of the settings in Listing 44.

5.5.1 lookForAlignDelims: `spacesBeforeAmpersand`

The `spacesBeforeAmpersand` can be specified in a few different ways. The basic form is demonstrated in Listing 46, but we can customise the behaviour further by specifying if we would like this value to change if it encounters a leading blank column; that is, when the first column contains only zero-width entries. We refer to this as the advanced form.

We demonstrate this feature in relation to Listing 59; upon running the following command

```bash
cmh:~ latexindent.pl aligned1.tex -o=+-default
```

then we receive the default output given in Listing 60.

<table>
<thead>
<tr>
<th>Listing 59: aligned1.tex</th>
<th>Listing 60: aligned1-default.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\begin{aligned}</code></td>
<td><code>\begin{aligned}</code></td>
</tr>
<tr>
<td>&amp; a &amp; b, <code>\</code></td>
<td>&amp; a &amp; b, <code>\</code></td>
</tr>
<tr>
<td>&amp; c &amp; d.</td>
<td>&amp; c &amp; d.</td>
</tr>
<tr>
<td><code>\end{aligned}</code></td>
<td><code>\end{aligned}</code></td>
</tr>
</tbody>
</table>

The settings in Listings 61 to 64 are all equivalent; we have used the not-yet discussed `noAdditionalIndent` field (see Section 5.8 on page 46) which will assist in the demonstration in what follows.

<table>
<thead>
<tr>
<th>Listing 61: sba1.yaml</th>
<th>Listing 62: sba2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>noAdditionalIndent</code>:</td>
<td><code>noAdditionalIndent</code>:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td><code>lookForAlignDelims</code>:</td>
<td><code>lookForAlignDelims</code>:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned:</td>
</tr>
<tr>
<td><code>spacesBeforeAmpersand</code>:</td>
<td><code>spacesBeforeAmpersand</code>:</td>
</tr>
<tr>
<td>default: 1</td>
<td>leadingBlankColumn: 1</td>
</tr>
</tbody>
</table>

Upon running the following commands

```bash
cmh:~ latexindent.pl aligned1.tex -l sba1.yaml
```

```bash
cmh:~ latexindent.pl aligned1.tex -l sba2.yaml
```

```bash
cmh:~ latexindent.pl aligned1.tex -l sba3.yaml
```

```bash
cmh:~ latexindent.pl aligned1.tex -l sba4.yaml
```

then we receive the (same) output given in Listing 65; we note that there is one space before each ampersand.
5.5 Aligning at delimiters

LISTING 65: aligned1-mod1.tex
\begin{aligned}
& a & b, \\
& c & d. \\
\end{aligned}

We note in particular:

- Listing 61 demonstrates the basic form for \texttt{lookForAlignDelims}; in this case, the default values are specified as in Listing 42 on page 27;
- Listing 62 demonstrates the advanced form for \texttt{lookForAlignDelims} and specified \texttt{spacesBeforeAmpersand}. The default value is 1;
- Listing 63 demonstrates the new advanced way to specify \texttt{spacesBeforeAmpersand}, and for us to set the default value that sets the number of spaces before ampersands which are not in leading blank columns. The default value is 1.

We note that \texttt{leadingBlankColumn} has not been specified in Listing 63, and it will inherit the value from default;

- Listing 64 demonstrates spaces to be used before amperands for leading blank columns. We note that \texttt{default} has not been specified, and it will be set to 1 by default.

We can customise the space before the ampersand in the leading blank column of Listing 65 by using either of Listings 66 and 67, which are equivalent.

<table>
<thead>
<tr>
<th>LISTING 66: sba5.yaml</th>
<th>LISTING 67: sba6.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned:</td>
<td>aligned:</td>
</tr>
<tr>
<td>spacesBeforeAmpersand:</td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td>leadingBlankColumn: 0</td>
<td>leadingBlankColumn: 0</td>
</tr>
<tr>
<td>default: 1</td>
<td>default: 1</td>
</tr>
</tbody>
</table>

Upon running

```
cmh:~$ latexindent.pl aligned1.tex -l sba5.yaml

```

then we receive the (same) output given in Listing 68. We note that the space before the ampersand in the leading blank column has been set to 0 by Listing 67.

We can demonstrated this feature further using the settings in Listing 70 which give the output in Listing 69.

<table>
<thead>
<tr>
<th>LISTING 68: aligned1-mod5.tex</th>
<th>LISTING 69: aligned1.tex using Listing 70</th>
<th>LISTING 70: sba7.yaml</th>
</tr>
</thead>
</table>
| \begin{aligned}
  & a & b, \\
  & c & d. \\
\end{aligned} | \begin{aligned}
  & a & b, \\
  & c & d. \\
\end{aligned} | noAdditionalIndent: 
  aligned: 1  
  lookForAlignDelims: 
  aligned: 
  spacesBeforeAmpersand: 
    leadingBlankColumn: 3  
  default: 0 |

5.5.2 \texttt{lookForAlignDelims}: alignFinalDoubleBackSlash

We explore the \texttt{alignFinalDoubleBackSlash} feature by using the file in Listing 71. Upon running the following commands

```
cmh:~$ latexindent.pl aligned1.tex -l sba7.yaml
```

\[ \text{Listing 68: aligned1-mod5.tex} \]
\[ \text{Listing 69: aligned1.tex using Listing 70} \]
\[ \text{Listing 70: sba7.yaml} \]
then we receive the respective outputs given in Listing 72 and Listing 73.

We note that in:

- Listing 72, by default, the first set of double backslashes in the first row of the \texttt{tabular} environment have been used for alignment;
- Listing 73, the final set of double backslashes in the first row have been used, because we specified alignFinalDoubleBackSlash as 1.

As of Version 3.0, the alignment routine works on mandatory and optional arguments within commands, and also within 'special' code blocks (see \texttt{specialBeginEnd} on page 38); for example, assuming that you have a command called \texttt{matrix} and that it is populated within \texttt{lookForAlignDelims} (which it is, by default), and that you run the command

\begin{verbatim}
cmh:$ latexindent.pl matrix1.tex
\end{verbatim}

then the before-and-after results shown in Listings 74 and 75 are achievable by default.

If you have blocks of code that you wish to align at the & character that are not wrapped in, for example, \texttt{\begin{tabular}...\end{tabular}}, then you can use the mark up illustrated in Listing 76; the default output is shown in Listing 77. Note that the %* must be next to each other, but that there can be any number of spaces (possibly none) between the * and \texttt{\begin{tabular}}; note also that you may use any environment name that you have specified in \texttt{lookForAlignDelims}.

With reference to Table 2 on page 45 and the, yet undiscussed, fields of noAdditionalIndent and indentRules (see Section 5.8 on page 46), these comment-marked blocks are considered environments.

\subsection{lookForAlignDelims: the dontMeasure feature}

The \texttt{lookForAlignDelims} field can, optionally, receive the dontMeasure option which can be specified in a few different ways. We will explore this feature in relation to the code given in Listing 78; the default output is shown in Listing 79.
The `dontMeasure` field can be specified as `largest`, and in which case, the largest element will not be measured; with reference to the YAML file given in Listing 81, we can run the command

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure1.yaml
```

and receive the output given in Listing 80.

```
\begin{tabular}{cccc}
  aaaaaa & bbbbb & ccc & dd \\
  11 & 2 & 33 & 4 \\
  5 & 66 & 7 & 8
\end{tabular}
```

We note that the `largest` column entries have not contributed to the measuring routine.

The `dontMeasure` field can also be specified in the form demonstrated in Listing 83. On running the following commands,

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure2.yaml
```

we receive the output in Listing 82.

```
\begin{tabular}{cccc}
  aaaaaa & bbbbb & ccc & dd \\
  11 & 2 & 33 & 4 \\
  5 & 66 & 7 & 8
\end{tabular}
```

We note that in Listing 83 we have specified entries not to be measured, one entry per line.

The `dontMeasure` field can also be specified in the forms demonstrated in Listing 85 and Listing 86. Upon running the commands

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure3.yaml
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure4.yaml
```

we receive the output given in Listing 84.
5.5 Aligning at delimiters

5.5.4 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature

The delimiter alignment will, by default, align code blocks at the ampersand character. The behaviour is controlled by the delimiterRegEx field within lookForAlignDelims; the default value is '(?<!\)(&)', which can be read as: an ampersand, as long as it is not immediately preceded by a backslash.

We note that in:

- Listing 85 we have specified entries not to be measured, each one has a string in the this field, together with an optional specification of applyTo as cell;
- Listing 86 we have specified entries not to be measured as a regular expression using the regex field, together with an optional specification of applyTo as cell field, together with an optional specification of applyTo as cell.

In both cases, the default value of applyTo is cell, and does not need to be specified.

We may also specify the applyTo field as row, a demonstration of which is given in Listing 88; upon running

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure5.yaml
```

we receive the output in Listing 87.

Finally, the applyTo field can be specified as row, together with a regex expression. For example, for the settings given in Listing 90, upon running

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure6.yaml
```

we receive the output in Listing 89.
5.5 Aligning at delimiters

**Warning!**

Important: note the ‘capturing’ parenthesis in the (&) which are necessary; if you intend to customise this field, then be sure to include them appropriately.

We demonstrate how to customise this with respect to the code given in Listing 91; the default output from latexindent.pl is given in Listing 92.

**Listing 91: tabbing.tex**

```latex
\begin{tabbing}
  aa \= bb \= cc \= dd \= ee \\
  \> 2 \> 1 \> 7 \> 3 \\
  \> 3 \> 2 \> 8 \> 3 \\
  \> 4 \> 2 \\
\end{tabbing}
```

**Listing 92: tabbing.tex default output**

```latex
\begin{tabbing}
  aa \= bb \= cc \= dd \= ee \\
  \> 2 \> 1 \> 7 \> 3 \\
  \> 3 \> 2 \> 8 \> 3 \\
  \> 4 \> 2 \\
\end{tabbing}
```

Let’s say that we wish to align the code at either the \= or \>. We employ the settings given in Listing 94 and run the command

```
cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx1.yaml
```

to receive the output given in Listing 93.

**Listing 93: tabbing.tex using Listing 94**

```latex
\begin{tabbing}
  aa \= bb \= cc \= dd \= ee \\
  \> 2 \> 1 \> 7 \> 3 \\
  \> 3 \> 2 \> 8 \> 3 \\
  \> 4 \> 2 \\
\end{tabbing}
```

**Listing 94: delimiterRegEx1.yaml**

```yaml
lookForAlignDelims:
  tabbing:
    delimiterRegEx: '(\(?:=|>)')
```

We note that:

- in Listing 93 the code has been aligned, as intended, at both the \= and \>;
- in Listing 94 we have heeded the warning and captured the expression using grouping parenthesis, specified a backslash using \ and said that it must be followed by either = or >.

We can explore delimiterRegEx a little further using the settings in Listing 96 and run the command

```
cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx2.yaml
```

to receive the output given in Listing 95.

**Listing 95: tabbing.tex using Listing 96**

```latex
\begin{tabbing}
  aa \= bb \= cc \= dd \= ee \\
  \> 2 \> 1 \> 7 \> 3 \\
  \> 3 \> 2 \> 8 \> 3 \\
  \> 4 \> 2 \\
\end{tabbing}
```

**Listing 96: delimiterRegEx2.yaml**

```yaml
lookForAlignDelims:
  tabbing:
    delimiterRegEx: '\(\\?)\>\)'
```

We note that only the \> have been aligned.

Of course, the other lookForAlignDelims options can be used alongside the delimiterRegEx; regardless of the type of delimiter being used (ampersand or anything else), the fields from Listing 42 on page 27 remain the same; for example, using the settings in Listing 98, and running
5.5 Aligning at delimiters

```latex
\begin{tabbing}
aa\=bb\=cc\=dd\=ee \hspace{2}\hspace{1}\hspace{7}\hspace{3} \\
\>2 \>1 \>7 \>3 \\
\>3 \>2 \>8 \>3 \\
\>4 \>2 \\
\end{tabbing}
```

LISTING 97: tabbing.tex using Listing 98

```yaml
lookForAlignDelims:
  tabbing:
    delimiterRegEx: '(\(?:=|>)')
    spacesBeforeAmpersand: 0
    spacesAfterAmpersand: 0
```

LISTING 98: delimiterRegEx3.yaml

It is possible that delimiters specified within delimiterRegEx can be of different lengths. Consider the file in Listing 99, and associated YAML in Listing 101. Note that the Listing 101 specifies the option for the delimiter to be either # or >, which are different lengths. Upon running the command

```bash
cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx4.yaml -o=+-mod4
```

we receive the output in Listing 100.

```latex
\begin{tabbing}
1 # 22 \> 333 \\
xxx # aaa # yyyyy \\
. # # & \\
\end{tabbing}
```

LISTING 99: tabbing1.tex

LISTING 100: tabbing1-mod4.tex

LISTING 101: delimiterRegEx4.yaml

You can set the delimiter justification as either left (default) or right, which will only have effect when delimiters in the same column have different lengths. Using the settings in Listing 103 and running the command

```bash
cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx5.yaml -o=+-mod5
```

gives the output in Listing 102.

```latex
\begin{tabbing}
1 # 22 \> 333 \\
xxx # aaa # yyyyy \\
. # # & \\
\end{tabbing}
```

LISTING 102: tabbing1-mod5.tex

LISTING 103: delimiterRegEx5.yaml

Note that in Listing 102 the second set of delimiters have been right aligned – it is quite subtle!

5.5.5 lookForAlignDelims: lookForChildCodeBlocks

There may be scenarios in which you would prefer to instruct latexindent.pl not to search for child blocks; in which case setting lookForChildCodeBlocks to 0 may be a good way to proceed. Using the settings from Listing 81 on page 34 on the file in Listing 104 and running the command

```bash
cmh:~$ latexindent.pl tabular-DM-1.tex -l=dontMeasure1.yaml -o=+-mod1
```

gives the output in Listing 105.
5.6 Indent after items, specials and headings

We can improve the output from Listing 105 by employing the settings in Listing 107

cmh:\$ latexindent.pl tabular-DM-1.tex -l=dontMeasure1a.yaml -o=+-mod1a

which gives the output in Listing 107.

5.6 Indent after items, specials and headings

The environment names specified in indentAfterItems tell latexindent.pl to look for \item commands; if these switches are set to 1 then indentation will be performed so as indent the code after each item. A demonstration is given in Listings 109 and 110

If you have your own item commands (perhaps you prefer to use myitem, for example) then you can put populate them in itemNames. For example, users of the exam document class might like to add parts to indentAfterItems and part to itemNames to their user settings (see Section 4 on page 16 for details of how to configure user settings, and Listing 13 on page 17 in particular.)

The fields specified in specialBeginEnd are, in their default state, focused on math mode begin and end statements, but there is no requirement for this to be the case; Listing 112 shows the default settings of specialBeginEnd.
5.6 Indent after items, specials and headings

The field `displayMath` represents `\[...\]`, `inlineMath` represents `$...$` and `displayMathTeX` represents `$$...$$`. You can, of course, rename these in your own YAML files (see Section 4.2 on page 17); indeed, you might like to set up your own special begin and end statements.

A demonstration of the before-and-after results are shown in Listings 113 and 114.

For each field, `lookForThis` is set to 1 by default, which means that `latexindent.pl` will look for this pattern; you can tell `latexindent.pl` not to look for the pattern, by setting `lookForThis` to 0.

There are examples in which it is advantageous to search for `specialBeginEnd` fields before searching for commands, and the `specialBeforeCommand` switch controls this behaviour. For example, consider the file shown in Listing 115.

Now consider the YAML files shown in Listings 116 and 117.

Upon running the following commands

```bash
[git] • main @ 45f6411 • 2022-01-02 • ☢ • V3.13.5
```
we receive the respective outputs in Listings 118 and 119.

Notice that in:

• Listing 118 the \left has been treated as a command, with one optional argument;
• Listing 119 the specialBeginEnd pattern in Listing 116 has been obeyed because Listing 117 specifies that the specialBeginEnd should be sought before commands.

You can, optionally, specify the middle field for anything that you specify in specialBeginEnd. For example, let’s consider the .tex file in Listing 120.

Upon saving the YAML settings in Listings 121 and 123 and running the commands

then we obtain the output given in Listings 122 and 124.
5.6 Indent after items, specials and headings

Listing 123: middle1.yaml

```yaml
specialBeginEnd:
  If:
    begin: '\If'
    middle:
      - '\ElsIf'
      - '\Else'
    end: '\EndIf'
  lookForThis: 1
```

Listing 124: special2.tex using Listing 123

```latex
\If
  something 0
\ElsIf
  something 1
\ElsIf
  something 2
\ElsIf
  something 3
\Else
  something 4
\EndIf
```

We note that:

- in Listing 122 the bodies of each of the Elsif statements have been indented appropriately;
- the Else statement has not been indented appropriately in Listing 122 – read on!
- we have specified multiple settings for the middle field using the syntax demonstrated in Listing 123 so that the body of the Else statement has been indented appropriately in Listing 124.

You may specify fields in specialBeginEnd to be treated as verbatim code blocks by changing lookForThis to be verbatim.

For example, beginning with the code in Listing 126 and the YAML in Listing 125, and running

```
cmh:~$ latexindent.pl special3.tex -l=special-verb1
```

then the output in Listing 126 is unchanged.

Listing 125: special-verb1.yaml

```yaml
specialBeginEnd:
  displayMath:
    lookForThis: verbatim
```

Listing 126: special3.tex and output using Listing 125

```
\[
\text{special code blocks can be treated as verbatim}
\]
```

We can combine the specialBeginEnd with the lookForAlignDelims feature. We begin with the code in Listing 127.

Listing 127: special-align.tex

```latex
\begin{tikzpicture}
  \path (A) edge node {0,1,L}(B)
       edge node {1,1,R} (C)
  (B) edge [loop above]node {1,1,L}(B)
       edge node {0,1,L}(C)
  (C) edge node {0,1,L}(D)
       edge [bend left]node {1,0,R}(E)
  (D) edge [loop below]node {1,1,R}(D)
       edge node {0,1,R}(A)
  (E) edge [bend left] node {1,0,R} (A);
\end{tikzpicture}
```

Let's assume that our goal is to align the code at the edge and node text; we employ the code given in Listing 128 and run the command

```
cmh:~$ latexindent.pl special4.tex -l=special-align
```

then the output in Listing 128 is unchanged.

Listing 128: special-align.yaml

```yaml
specialBeginEnd:
  displayMath:
    lookForAlignDelims:
      alignDelims: \[
```

Listing 129: special-align.tex

```
\[
\text{special code blocks can be treated as verbatim}
\]
```
5.6 Indent after items, specials and headings

\begin{tikzpicture}
\path (A) edge node {0,1,L} (B)
edge node {1,1,R} (C)
\end{tikzpicture}

The output in Listing 129 is not quite ideal. We can tweak the settings within Listing 128 in order to improve the output; in particular, we employ the code in Listing 130 and run the command

\texttt{c mh:~$ latexindent.pl special-align.tex -l edge-node2.yaml -o+=mod2}

to receive the output in Listing 131.

\begin{tikzpicture}
\path (A) edge node {0,1,L} (B)
edge node {1,1,R} (C)
\end{tikzpicture}

The \texttt{lookForThis} field can be considered optional; by default, it is assumed to be 1, which is demonstrated in Listing 130.

\textbf{indentAfterHeadings: }\texttt{\{fields\}}

This field enables the user to specify indentation rules that take effect after heading commands such as \texttt{\textbackslash part}, \texttt{\textbackslash chapter, \textbackslash section, \textbackslash subsection*}, or indeed any user-specified command written in this field.\footnote{There is a slight difference in interface for this field when comparing Version 2.2 to Version 3.0; see appendix H on page 148 for details.}

\texttt{\textbackslash begin\{tikzpicture\}
\path (A) edge node {0,1,L} (B)
edge node {1,1,R} (C)
\end{tikzpicture}
5.6 Indent after items, specials and headings

The default settings do not place indentation after a heading, but you can easily switch them on by changing `indentAfterThisHeading` from 0 to 1. The `level` field tells `latexindent.pl` the hierarchy of the heading structure in your document. You might, for example, like to have both `section` and `subsection` set with `level`: 3 because you do not want the indentation to go too deep.

You can add any of your own custom heading commands to this field, specifying the `level` as appropriate. You can also specify your own indentation in `indentRules` (see Section 5.8 on page 46); you will find the default `indentRules` contains `chapter: " "` which tells `latexindent.pl` simply to use a space character after chapter headings (once `indent` is set to 1 for chapter).

For example, assuming that you have the code in Listing 133 saved into `headings1.yaml`, and that you have the text from Listing 134 saved into `headings1.tex`.

If you run the command

```
cmh:~$ latexindent.pl headings1.tex -l=headings1.yaml
```

then you should receive the output given in Listing 135.

Now say that you modify the YAML from Listing 133 so that the paragraph `level` is 1; after running

```
cmh:~$ latexindent.pl headings1.tex -l=headings1.yaml
```
you should receive the code given in Listing 136; notice that the paragraph and subsection are at the same indentation level.

\begin{one}
\begin{two}
\begin{three}
\begin{four}
four
\end{four}
\end{three}
\end{two}
\end{one}

You can control the maximum indentation given to your file by specifying the \texttt{maximumIndentation} field as horizontal space (but \textit{not} including tabs). This feature uses the \texttt{Text::Tabs} module [28], and is \textit{off} by default.

For example, consider the example shown in Listing 137 together with the default output shown in Listing 138.

\begin{one}
\begin{two}
\begin{three}
\begin{four}
four
\end{four}
\end{three}
\end{two}
\end{one}

\begin{one}
___one
___\begin{two}
___two
___\begin{three}
___three
____\begin{four}
____four
____\end{four}
____\end{three}
___\end{two}
\end{one}

Now say that, for example, you have the \texttt{max-indentation1.yaml} from Listing 139 and that you run the following command:

cmh:~$ latexindent.pl mult-nested.tex -l=max-indentation1

You should receive the output shown in Listing 140.

\begin{one}
\begin{two}
\begin{three}
\begin{four}
four
\end{four}
\end{three}
\end{two}
\end{one}

\begin{one}
___one
___\begin{two}
___two
___\begin{three}
___three
____\begin{four}
____four
____\end{four}
____\end{three}
___\end{two}
\end{one}

Comparing the output in Listings 138 and 140 we notice that the (default) tabs of indentation have been replaced by a single space.

In general, when using the \texttt{maximumIndentation} feature, any leading tabs will be replaced by equivalent spaces except, of course, those found in \texttt{verbatimEnvironments} (see Listing 18 on page 22) or \texttt{noIndentBlock} (see Listing 24 on page 23).

5.7 The code blocks known latexindent.pl

As of Version 3.0, \texttt{latexindent.pl} processes documents using code blocks; each of these are shown in Table 2.
<table>
<thead>
<tr>
<th>Code block</th>
<th>characters allowed in name</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>environments</td>
<td>a-zA-Z*0-9__</td>
<td>\begin{myenv} body of myenv \end{myenv}</td>
</tr>
<tr>
<td>optionalArguments</td>
<td>inherits name from parent (e.g environment name)</td>
<td>[ opt arg text ]</td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>inherits name from parent (e.g environment name)</td>
<td>{ mand arg text }</td>
</tr>
<tr>
<td>commands</td>
<td>+a-zA-Z*0-9__:</td>
<td>\mycommand{arguments}</td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>a-zA-Z*0-9__/:\h{}#-</td>
<td>my \key/.style=(arguments)</td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>0-9_.a-zA-Z**&gt;&lt;</td>
<td>in{arguments}</td>
</tr>
<tr>
<td>UnNamedGroupingBracesBrackets</td>
<td>No name!</td>
<td>{ or \ or &amp; or ) or ( or $ followed by \arguments)</td>
</tr>
</tbody>
</table>
| ifElseFi                     | @a-zA-Z but must begin with either \if of \@if | \ifnum...
|                               |                            | \else
|                               |                            | \if
|                               |                            | \fi                                      |
| items                        | User specified, see Listings 108 and 111 on page 38 | \begin{enumerate} \item ...
|                               |                            | \end{enumerate}                           |
| specialBeginEnd              | User specified, see Listing 112 on page 39 | \[
|                               |                            |   ...
|                               |                            | \]                                      |
| afterHeading                 | User specified, see Listing 132 on page 43 | \chapter{title}
|                               |                            | \section{title}                           |
| filecontents                 | User specified, see Listing 34 on page 25 | \begin{filecontents} ...
|                               |                            | \end{filecontents}                        |
We will refer to these code blocks in what follows. Note that the fine tuning of the definition of the code blocks detailed in Table 2 is discussed in Section 9 on page 128.

5.8 noAdditionalIndent and indentRules

latexindent.pl operates on files by looking for code blocks, as detailed in Section 5.7 on page 44; for each type of code block in Table 2 on the preceding page (which we will call a \textit{thing}) in what follows it searches YAML fields for information in the following order:

1. noAdditionalIndent for the \textit{name} of the current \textit{thing};
2. indentRules for the \textit{name} of the current \textit{thing};
3. noAdditionalIndentGlobal for the \textit{type} of the current \textit{thing};
4. indentRulesGlobal for the \textit{type} of the current \textit{thing}.

Using the above list, the first piece of information to be found will be used; failing that, the value of defaultIndent is used. If information is found in multiple fields, the first one according to the list above will be used; for example, if information is present in both indentRules and in noAdditionalIndentGlobal, then the information from indentRules takes priority.

We now present details for the different type of code blocks known to latexindent.pl, as detailed in Table 2 on the previous page; for reference, there follows a list of the code blocks covered.

5.8.1 Environments and their arguments ........................................... 46
5.8.2 Environments with items ....................................................... 53
5.8.3 Commands with arguments ..................................................... 54
5.8.4 ifelsefi code blocks ............................................................. 56
5.8.5 specialBeginEnd code blocks ................................................ 57
5.8.6 afterHeading code blocks ...................................................... 58
5.8.7 The remaining code blocks .................................................... 60
  5.8.7.1 keyEqualsValuesBracesBrackets ....................................... 60
  5.8.7.2 namedGroupingBracesBrackets ....................................... 61
  5.8.7.3 UnNamedGroupingBracesBrackets .................................... 61
  5.8.7.4 filecontents ............................................................. 62
5.8.8 Summary ................................................................. 62

5.8.1 Environments and their arguments

There are a few different YAML switches governing the indentation of environments; let’s start with the code shown in Listing 141.

\begin{outer}
\begin{myenv}
  body of environment
  body of environment
  body of environment
\end{myenv}
\end{outer}

\textbf{noAdditionalIndent: \{fields\}}

If we do not wish \texttt{myenv} to receive any additional indentation, we have a few choices available to us, as demonstrated in Listings 142 and 143.
On applying either of the following commands,

```bash
cmh:~$ latexindent.pl myenv.tex -l myenv-noAdd1.yaml
cmh:~$ latexindent.pl myenv.tex -l myenv-noAdd2.yaml
```

we obtain the output given in Listing 144; note in particular that the environment `myenv` has not received any additional indentation, but that the outer environment has still received indentation.

Upon changing the YAML files to those shown in Listings 145 and 146, and running either

```bash
cmh:~$ latexindent.pl myenv.tex -l myenv-noAdd3.yaml
cmh:~$ latexindent.pl myenv.tex -l myenv-noAdd4.yaml
```

we obtain the output given in Listing 147.

Let’s now allow `myenv` to have some optional and mandatory arguments, as in Listing 148.
5.8 noAdditionalIndent and indentRules

Upon running

```
cmh:~$ latexindent.pl -l=myenv-noAdd1.yaml myenv-args.tex
```

we obtain the output shown in Listing 149; note that the optional argument, mandatory argument and body all have received no additional indent. This is because, when noAdditionalIndent is specified in ‘scalar’ form (as in Listing 142), then all parts of the environment (body, optional and mandatory arguments) are assumed to want no additional indent.

We may customise noAdditionalIndent for optional and mandatory arguments of the myenv environment, as shown in, for example, Listings 150 and 151.

Upon running

```
cmh:~$ latexindent.pl myevn.tex -l=myenv-noAdd5.yaml
cmh:~$ latexindent.pl myevn.tex -l=myenv-noAdd6.yaml
```

we obtain the respective outputs given in Listings 152 and 153. Note that in Listing 152 the text for the optional argument has not received any additional indentation, and that in Listing 153 the mandatory argument has not received any additional indentation; in both cases, the body has not received any additional indentation.
We may also specify indentation rules for environment code blocks using the \texttt{indentRules} field; see, for example, Listings 154 and 155.

On applying either of the following commands,

\begin{verbatim}
cmh:$ latexindent.pl myenv.tex -l myenv-rules1.yaml  
cmh:$ latexindent.pl myenv.tex -l myenv-rules2.yaml
\end{verbatim}

we obtain the output given in Listing 156; note in particular that the environment \texttt{myenv} has received one tab (from the \texttt{outer} environment) plus three spaces from Listing 154 or 155.

If you specify a field in \texttt{indentRules} using anything other than horizontal space, it will be ignored.

Returning to the example in Listing 148 that contains optional and mandatory arguments. Upon using Listing 154 as in

\begin{verbatim}
cmh:$ latexindent.pl myenv-args.tex -l=myenv-rules1.yaml
\end{verbatim}

we obtain the output in Listing 157; note that the body, optional argument and mandatory argument of \texttt{myenv} have \textit{all} received the same customised indentation.
You can specify different indentation rules for the different features using, for example, Listings 158 and 159.

After running

cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules3.yaml
cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules4.yaml

then we obtain the respective outputs given in Listings 160 and 161.

Note that in Listing 160, the optional argument has only received a single space of indentation, while the mandatory argument has received the default (tab) indentation; the environment body has received three spaces of indentation.

In Listing 161, the optional argument has received the default (tab) indentation, the mandatory argument has received two tabs of indentation, and the body has received three spaces of indentation.

Assuming that your environment name is not found within neither noAdditionalIndent nor indentRules, the next place that latexindent.pl will look is noAdditionalIndentGlobal, and in particular for the environments key (see Listing 162).
Let's say that you change the value of \texttt{environments} to 1 in Listing 162, and that you run

\begin{verbatim}
cmh:~$ latexindent.pl myenv-args.tex -l env-noAdditionalGlobal.yaml
cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-noAdditionalGlobal.yaml
\end{verbatim}

The respective output from these two commands are in Listings 163 and 164; in Listing 163 notice that both environments receive no additional indentation but that the arguments of \texttt{myenv} still do receive indentation. In Listing 164 notice that the outer environment does not receive additional indentation, but because of the settings from \texttt{myenv-rules1.yaml} (in Listing 154 on page 49), the \texttt{myenv} environment still does receive indentation.

\begin{verbatim}
\begin{outer}
\begin{myenv}[
  optional argument text
  optional argument text]
  { mandatory argument text
    mandatory argument text}
body of environment
body of environment
body of environment
\end{myenv}
\end{outer}
\end{verbatim}

In fact, \texttt{noAdditionalIndentGlobal} also contains keys that control the indentation of optional and mandatory arguments; on referencing Listings 165 and 166

\begin{verbatim}
\begin{outer}
\begin{myenv}[
  optional argument text
  optional argument text]
  { mandatory argument text
    mandatory argument text}
body of environment
body of environment
body of environment
\end{myenv}
\end{outer}
\end{verbatim}

we may run the commands

\begin{verbatim}
cmh:~$ latexindent.pl myenv-args.tex -local opt-args-no-add-glob.yaml
cmh:~$ latexindent.pl myenv-args.tex -local mand-args-no-add-glob.yaml
\end{verbatim}

which produces the respective outputs given in Listings 167 and 168. Notice that in Listing 167 the \textit{optional} argument has not received any additional indentation, and in Listing 168 the \textit{mandatory} argument has not received any additional indentation.
The final check that \texttt{latexindent.pl} will make is to look for \texttt{indentRulesGlobal} as detailed in Listing 169.

\begin{verbatim}
indentRulesGlobal: \{fields\}
\end{verbatim}

If you change the \texttt{environments} field to anything involving horizontal space, say " ", and then run the following commands

\texttt{cmh:~$ latexindent.pl myenv-args.tex -l env-indentRules.yaml}
\texttt{cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml, env-indentRules.yaml}

then the respective output is shown in Listings 170 and 171. Note that in Listing 170, both the environment blocks have received a single-space indentation, whereas in Listing 171 the outer environment has received single-space indentation (specified by \texttt{indentRulesGlobal}), but \texttt{myenv} has received " ", as specified by the particular \texttt{indentRules} for \texttt{myenv} Listing 154 on page 49.

You can specify \texttt{indentRulesGlobal} for both optional and mandatory arguments, as detailed in Listings 172 and 173

\begin{verbatim}
You can specify indentRulesGlobal for both optional and mandatory arguments, as detailed in Listings 172 and 173.
\end{verbatim}

Upon running the following commands

\begin{verbatim}
You can specify indentRulesGlobal for both optional and mandatory arguments, as detailed in Listings 172 and 173.
\end{verbatim}
we obtain the respective outputs in Listings 174 and 175. Note that the optional argument in Listing 174 has received two tabs worth of indentation, while the mandatory argument has done so in Listing 175.

```
\begin{outer}
  \begin{myenv}[]
    \begin{itemize}
      \item some text here
      some more text here
      \item another item
      some more text here
    \end{itemize}
  \end{myenv}
\end{outer}
```

```
\begin{outer}
  \begin{myenv}[]
    \begin{itemize}
      \item some text here
      some more text here
      \item another item
      some more text here
    \end{itemize}
  \end{myenv}
\end{outer}
```

5.8.2 Environments with items

With reference to Listings 108 and 111 on page 38, some commands may contain item commands; for the purposes of this discussion, we will use the code from Listing 109 on page 38.

Assuming that you've populated itemNames with the name of your item, you can put the item name into noAdditionalIndent as in Listing 176, although a more efficient approach may be to change the relevant field in itemNames to 0. Similarly, you can customise the indentation that your item receives using indentRules, as in Listing 177

```
noAdditionalIndent:
  item: 1

indentRules:
  item: " "
```

Upon running the following commands

```
cmh:~$ latexindent.pl items1.tex -local item-noAdd1.yaml
cmh:~$ latexindent.pl items1.tex -local item-rules1.yaml
```

the respective outputs are given in Listings 178 and 179; note that in Listing 178 that the text after each item has not received any additional indentation, and in Listing 179, the text after each item has received a single space of indentation, specified by Listing 177.

```
\begin{itemize}
  \item some text here
  some more text here
  \item another item
  some more text here
\end{itemize}
```

```
\begin{itemize}
  \item some text here
  some more text here
  \item another item
  some more text here
\end{itemize}
```

Alternatively, you might like to populate noAdditionalIndentGlobal or indentRulesGlobal using the items key, as demonstrated in Listings 180 and 181. Note that there is a need to 'reset/remove' the item field from indentRules in both cases (see the hierarchy description given on page 46) as the item command is a member of indentRules by default.
5.8 noAdditionalIndent and indentRules

Upon running the following commands,

```bash
cmh:~$ latexindent.pl items1.tex -local items-noAdditionalGlobal.yaml
cmh:~$ latexindent.pl items1.tex -local items-indentRulesGlobal.yaml
```

the respective outputs from Listings 178 and 179 are obtained; note, however, that all such item commands without their own individual noAdditionalIndent or indentRules settings would behave as in these listings.

5.8.3 Commands with arguments

Let’s begin with the simple example in Listing 182; when latexindent.pl operates on this file, the default output is shown in Listing 183.

```latex
\mycommand
{   mand arg text
mand arg text}
[   opt arg text
opt arg text ]
```

As in the environment-based case (see Listings 142 and 143 on page 47) we may specify noAdditionalIndent either in ‘scalar’ form, or in ‘field’ form, as shown in Listings 184 and 185.

```yaml
# Listing 184: mycommand-noAdd1.yaml
noAdditionalIndent:
  mycommand: 1
```

```yaml
# Listing 185: mycommand-noAdd2.yaml
noAdditionalIndent:
  mycommand: body: 1
```

After running the following commands,

```bash
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd1.yaml
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd2.yaml
```

we receive the respective output given in Listings 186 and 187.

---

6The command code blocks have quite a few subtleties, described in Section 5.9 on page 62.
Note that in Listing 186 that the ‘body’, optional argument and mandatory argument have all received no additional indentation, while in Listing 187, only the ‘body’ has not received any additional indentation. We define the ‘body’ of a command as any lines following the command name that include its optional or mandatory arguments.

We may further customise noAdditionalIndent for mycommand as we did in Listings 150 and 151 on page 48; explicit examples are given in Listings 188 and 189.

After running the following commands,

cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd3.yaml

cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd4.yaml

we receive the respective output given in Listings 190 and 191.

Attentive readers will note that the body of mycommand in both Listings 190 and 191 has received no additional indent, even though body is explicitly set to 0 in both Listings 188 and 189. This is because, by default, noAdditionalIndentGlobal for commands is set to 1 by default; this can be easily fixed as in Listings 192 and 193.
After running the following commands,

```
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd5.yaml
```

we receive the respective output given in Listings 194 and 195.

**Listing 194:** mycommand.tex using Listing 192

```
\mycommand
  {  
    mand arg text
    mand arg text
  } [  
    opt arg text
    opt arg text
  ]
```

**Listing 195:** mycommand.tex using Listing 193

```
\mycommand
  {  
    mand arg text
    mand arg text
  } [  
    opt arg text
    opt arg text
  ]
```

Both indentRules and indentRulesGlobal can be adjusted as they were for environment code blocks, as in Listings 158 and 159 on page 50 and Listings 169, 172 and 173 on page 52.

### 5.8.4 ifelsefi code blocks

Let's use the simple example shown in Listing 196; when latexindent.pl operates on this file, the output as in Listing 197; note that the body of each of the \if statements have been indented, and that the \else statement has been accounted for correctly.

**Listing 196:** ifelsefi1.tex

```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

**Listing 197:** ifelsefi1.tex default output

```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

It is recommended to specify noAdditionalIndent and indentRules in the ‘scalar’ form only for these type of code blocks, although the ‘field’ form would work, assuming that body was specified. Examples are shown in Listings 198 and 199.

**Listing 198:** ifnum-noAdd.yaml

```
noAdditionalIndent:
  ifnum: 1
```

**Listing 199:** ifnum-indent-rules.yaml

```
indentRules:
  ifnum: " ">
```

After running the following commands,

```
cmh:~$ latexindent.pl ifelsefi1.tex -local ifnum-noAdd.yaml
```

we receive the respective output given in Listings 200 and 201; note that in Listing 200, the ifnum code block has not received any additional indentation, while in Listing 201, the ifnum code block has received one tab and two spaces of indentation.
5.8 noAdditionalIndent and indentRules

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 202 and 203.

Upon running the following commands

```
$ latexindent ifelsefi1.tex -local ifelsefi-noAdd-glob.yaml
$ latexindent ifelsefi1.tex -l ifelsefi-indent-rules-global.yaml
```

we receive the outputs in Listings 204 and 205; notice that in Listing 204 neither of the ifelsefi code blocks have received indentation, while in Listing 205 both code blocks have received a single space of indentation.

We can further explore the treatment of ifElseFi code blocks in Listing 206, and the associated default output given in Listing 207; note, in particular, that the bodies of each of the 'or statements' have been indented.

5.8.5 specialBeginEnd code blocks

Let's use the example from Listing 113 on page 39 which has default output shown in Listing 114 on page 39.

It is recommended to specify noAdditionalIndent and indentRules in the 'scalar' form for these type of code blocks, although the 'field' form would work, assuming that body was specified. Examples are shown in Listings 208 and 209.
After running the following commands,

\begin{verbatim}
cmh:~$ latexindent.pl special1.tex -local displayMath-noAdd.yaml
cmh:~$ latexindent.pl special1.tex -l displayMath-indent-rules.yaml
\end{verbatim}

we receive the respective output given in Listings 210 and 211; note that in Listing 210, the displayMath code block has not received any additional indentation, while in Listing 211, the displayMath code block has received three tabs worth of indentation.

\begin{verbatim}
LISTING 208: displayMath-noAdd.yaml
noAdditionalIndent:
  displayMath: 1

LISTING 209: displayMath-indent-rules.yaml
indentRules:
  displayMath: "\t\t\t"
\end{verbatim}

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 212 and 213.

\begin{verbatim}
LISTING 210: special1.tex using Listing 208
The function $f$ has formula
\[
  f(x)=x^2.
\]
If you like splitting dollars,
$ 
  g(x)=f(2x)
$

LISTING 211: special1.tex using Listing 209
The function $f$ has formula
\[
  f(x)=x^2.
\]
If you like splitting dollars,
$ 
  g(x)=f(2x)
$
\end{verbatim}

Upon running the following commands

\begin{verbatim}
cmh:~$ latexindent.pl special1.tex -local special-noAdd-glob.yaml
cmh:~$ latexindent.pl special1.tex -l special-indent-rules-global.yaml
\end{verbatim}

we receive the outputs in Listings 214 and 215; notice that in Listing 214 neither of the special code blocks have received indentation, while in Listing 215 both code blocks have received a single space of indentation.

\begin{verbatim}
LISTING 212: special-noAdd-glob.yaml
noAdditionalIndentGlobal:
  specialBeginEnd: 1

LISTING 213: special-indent-rules-global.yaml
indentRulesGlobal:
  specialBeginEnd: " ">
\end{verbatim}

Let's use the example Listing 216 for demonstration throughout this Section. As discussed on page 43, by default latexindent.pl will not add indentation after headings.
5.8 noAdditionalIndent and indentRules

\begin{Verbatim}
\begin{minipage}{0.6\textwidth}
\begin{yaml}
indentAfterHeadings:
  paragraph:
    indentAfterThisHeading: 1
    level: 1
\end{yaml}
\end{minipage}
\end{Verbatim}

On using the YAML file in Listing 218 by running the command

\texttt{cmh:~}\$ \texttt{latexindent.pl headings2.tex \-l headings3.yaml}

we obtain the output in Listing 217. Note that the argument of paragraph has received (default) indentation, and that the body after the heading statement has received (default) indentation.

\begin{Verbatim}
\begin{minipage}{0.6\textwidth}
\begin{yaml}
indentAfterHeadings:
  paragraph:
    indentAfterThisHeading: 1
    level: 1
\end{yaml}
\end{minipage}
\end{Verbatim}

If we specify noAdditionalIndent as in Listing 220 and run the command

\texttt{cmh:~}\$ \texttt{latexindent.pl headings2.tex \-l headings4.yaml}

then we receive the output in Listing 219. Note that the arguments and the body after the heading of paragraph has received no additional indentation, because we have specified noAdditionalIndent in scalar form.

\begin{Verbatim}
\begin{minipage}{0.6\textwidth}
\begin{yaml}
indentAfterHeadings:
  paragraph:
    indentAfterThisHeading: 1
    level: 1
noAdditionalIndent:
  paragraph: 1
\end{yaml}
\end{minipage}
\end{Verbatim}

Similarly, if we specify indentRules as in Listing 222 and run analogous commands to those above, we receive the output in Listing 221; note that the body, mandatory argument and content after the heading of paragraph have all received three tabs worth of indentation.

\begin{Verbatim}
\begin{minipage}{0.6\textwidth}
\begin{yaml}
indentAfterHeadings:
  paragraph:
    indentAfterThisHeading: 1
    level: 1
indentRules:
  paragraph: "\t\t\t"
\end{yaml}
\end{minipage}
\end{Verbatim}

We may, instead, specify noAdditionalIndent in ‘field’ form, as in Listing 224 which gives the output in Listing 223.
Analogously, we may specify indentRules as in Listing 226 which gives the output in Listing 225; note that mandatory argument text has only received a single space of indentation, while the body after the heading has received three tabs worth of indentation.

Finally, let's consider noAdditionalIndentGlobal and indentRulesGlobal shown in Listings 228 and 230 respectively, with respective output in Listings 227 and 229. Note that in Listing 228 the mandatory argument of paragraph has received a (default) tab's worth of indentation, while the body after the heading has received no additional indentation. Similarly, in Listing 229, the argument has received both a (default) tab plus two spaces of indentation (from the global rule specified in Listing 230), and the remaining body after paragraph has received just two spaces of indentation.

5.8.7 The remaining code blocks

Referencing the different types of code blocks in Table 2 on page 45, we have a few code blocks yet to cover; these are very similar to the commands code block type covered comprehensively in Section 5.8.3 on page 54, but a small discussion defining these remaining code blocks is necessary.

5.8.7.1 keyEqualsValuesBracesBrackets

latexindent.pl defines this type of code block by the following criteria:

- it must immediately follow either { OR [ OR , with comments and blank lines allowed.
- then it has a name made up of the characters detailed in Table 2 on page 45;
• then an `=` symbol;
• then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the `keyEqualsValuesBracesBrackets:` follow and `keyEqualsValuesBracesBrackets:` name fields of the fine tuning section in Listing 521 on page 128

An example is shown in Listing 231, with the default output given in Listing 232.

<table>
<thead>
<tr>
<th>Listing 231: pgfkeys1.tex</th>
<th>Listing 232: pgfkeys1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\pgfkeys{/tikz/.cd,</code></td>
<td><code>\pgfkeys{/tikz/.cd,</code></td>
</tr>
<tr>
<td><code>start coordinate/.initial={0,</code></td>
<td><code>start coordinate/.initial={0,</code></td>
</tr>
<tr>
<td><code>\vertfactor},</code></td>
<td><code>\vertfactor},</code></td>
</tr>
<tr>
<td>}</td>
<td><code>}</code></td>
</tr>
</tbody>
</table>

In Listing 232, note that the maximum indentation is three tabs, and these come from:

• the `\pgfkeys` command’s mandatory argument;
• the `start coordinate/.initial` key’s mandatory argument;
• the `start coordinate/.initial` key’s body, which is defined as any lines following the name of the key that include its arguments. This is the part controlled by the `body` field for `noAdditionalIndent` and friends from page 46.

5.8.7.2 namedGroupingBracesBrackets

This type of code block is mostly motivated by tikz-based code; we define this code block as follows:

• it must immediately follow either `horizontal space` OR `one or more line breaks` OR `{` OR `\[` OR `\)` OR `(` OR `\$`; 
• the name may contain the characters detailed in Table 2 on page 45;
• then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the `NamedGroupingBracesBrackets:` follow and `NamedGroupingBracesBrackets:` name fields of the fine tuning section in Listing 521 on page 128

A simple example is given in Listing 233, with default output in Listing 234.

<table>
<thead>
<tr>
<th>Listing 233: child1.tex</th>
<th>Listing 234: child1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\coordinate</code></td>
<td><code>\coordinate</code></td>
</tr>
<tr>
<td><code>child\[grow=down\]{</code></td>
<td><code>child\[grow=down\]{</code></td>
</tr>
<tr>
<td><code>edge from parent [antiparticle]</code></td>
<td><code>edge from parent [antiparticle]</code></td>
</tr>
<tr>
<td><code>node [above=3pt] \{\$C\}$</code></td>
<td><code>node [above=3pt] \{\$C\}$</code></td>
</tr>
<tr>
<td>}</td>
<td><code>}</code></td>
</tr>
</tbody>
</table>

In particular, `latexindent.pl` considers `child`, `parent` and `node` all to be `namedGroupingBracesBrackets`\(^7\). Referencing Listing 234, note that the maximum indentation is two tabs, and these come from:

• the child’s mandatory argument;
• the child’s body, which is defined as any lines following the name of the `namedGroupingBracesBrackets` that include its arguments. This is the part controlled by the `body` field for `noAdditionalIndent` and friends from page 46.

5.8.7.3 UnNamedGroupingBracesBrackets

occur in a variety of situations; specifically, we define this type of code block as satisfying the following criteria:

• it must immediately follow either `{` OR `\[`, `OR \&` OR `\)` OR ( OR `\$`;

\(^7\)You may like to verify this by using the `-tt` option and checking `indent.log`!
5.9 Commands and the strings between their arguments

The command code blocks will always look for optional (square bracketed) and mandatory (curly braced) arguments which can contain comments, line breaks and ‘beamer’ commands 〈.*?〉 between them. There are switches that can allow them to contain other strings, which we discuss next.

The commandCodeBlocks field contains a few switches detailed in Listing 239.
5.9 Commands and the strings between their arguments

The need for this field was mostly motivated by commands found in code used to generate images in PSTricks and \textit{tikz}; for example, let's consider the code given in Listing 240.

\begin{verbatim}
defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))\cos(v+\Pi)}
{(2+\cos(u))\sin(v+\Pi)}
{\sin(u)}
\end{verbatim}

Notice that the \texttt{defFunction} command has an optional argument, followed by a mandatory argument, followed by a round-parenthesis argument, \((u,v)\).

By default, because \texttt{roundParenthesesAllowed} is set to 1 in Listing 239, then \texttt{latexindent.pl} will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 240, \texttt{latexindent.pl} finds all the arguments of \texttt{defFunction}, both before and after \((u,v)\).

The default output from running \texttt{latexindent.pl} on Listing 240 actually leaves it unchanged (see Listing 241); note in particular, this is because of \texttt{noAdditionalIndentGlobal} as discussed on page 55.

Upon using the YAML settings in Listing 243, and running the command

\begin{verbatim}
cmh:~$ latexindent.pl pstricks1.tex -l noRoundParentheses.yaml
\end{verbatim}

we obtain the output given in Listing 242.

\begin{verbatim}
defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))\cos(v+\Pi)}
{(2+\cos(u))\sin(v+\Pi)}
{\sin(u)}
\end{verbatim}

Notice the difference between Listing 241 and Listing 242; in particular, in Listing 242, because round parentheses are not allowed, \texttt{latexindent.pl} finds that the \texttt{defFunction} command finishes at the first opening round parenthesis. As such, the remaining braced, mandatory, arguments are found to be \texttt{UnNamedGroupingBracesBrackets} (see Table 2 on page 45) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 242.

Let's explore this using the YAML given in Listing 245 and run the command

\begin{verbatim}
cmh:~$ latexindent.pl pstricks1.tex -l Listing 245.yaml
\end{verbatim}

which results in the output given in Listing 246.

\begin{verbatim}
defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))\cos(v+\Pi)}
{(2+\cos(u))\sin(v+\Pi)}
{\sin(u)}
\end{verbatim}

Notice the difference between Listing 242 and Listing 246; in particular, in Listing 246, because round parentheses are allowed, \texttt{latexindent.pl} finds that the \texttt{defFunction} command finishes at the first opening round parenthesis. As such, the remaining braced, mandatory, arguments are found to be \texttt{UnNamedGroupingBracesBrackets} (see Table 2 on page 45) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 246.
5.9 Commands and the strings between their arguments

Then the output is as in Listing 244.

Then the output is as in Listing 244.

Listing 244: \texttt{pstricks1.tex} using Listing 245
\begin{verbatim}
defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))*\cos(v+\Pi)}
{(2+\cos(u))*\sin(v+\Pi)}
{\sin(u)}
\end{verbatim}

Notice in Listing 244 that the body of the \texttt{defFunction} command i.e, the subsequent lines containing arguments after the command name, have received the single space of indentation specified by Listing 245.

Strings Allowed Between Arguments: \texttt{(fields)}

Tikz users may well specify code such as that given in Listing 246; processing this code using latexindent.pl gives the default output in Listing 247.

Listing 246: \texttt{tikz-node1.tex}
\begin{verbatim}
draw[thin]
(c) to[in=110,out=-90]
++(0,-0.5cm)
node[below,align=left,scale=0.5]
\end{verbatim}

With reference to Listing 239 on the previous page, we see that the strings
to, node, ++
are all allowed to appear between arguments; importantly, you are encouraged to add further names to this field as necessary. This means that when latexindent.pl processes Listing 246, it consumes:

- the optional argument \texttt{[thin]}
- the round-bracketed argument \texttt{(c)} because \texttt{roundParenthesesAllowed} is 1 by default
- the string \texttt{to} (specified in \texttt{stringsAllowedBetweenArguments})
- the optional argument \texttt{[in=110,out=-90]}
- the string \texttt{++} (specified in \texttt{stringsAllowedBetweenArguments})
- the round-bracketed argument \texttt{(0,-0.5cm)} because \texttt{roundParenthesesAllowed} is 1 by default
- the string \texttt{node} (specified in \texttt{stringsAllowedBetweenArguments})
- the optional argument \texttt{[below,align=left,scale=0.5]}

We can explore this further, for example using Listing 249 and running the command

\begin{verbatim}
cmh:~$ latexindent.pl tikz-node1.tex -l draw.yaml
\end{verbatim}

we receive the output given in Listing 248.
5.9 Commands and the strings between their arguments

LISTING 248: tikz-node1.tex using Listing 249

\draw[thin]
\NodeToInOut{(c)}{in=110,out=-90}
++(0,-0.5cm)
\Node{below,align=left,scale=0.5}

Notice that each line after the \draw command (its 'body') in Listing 248 has been given the appropriate two-spaces worth of indentation specified in Listing 249.

Let's compare this with the output from using the YAML settings in Listing 251, and running the command

cmh:~$ latexindent.pl tikz-node1.tex -l no-strings.yaml

given in Listing 250.

LISTING 250: tikz-node1.tex using Listing 251

\draw[thin]
NodeToInOut{(c)}{in=110,out=-90}
++(0,-0.5cm)
Node{below,align=left,scale=0.5}

In this case, latexindent.pl sees that:

- the \draw command finishes after the (c), as stringsAllowedBetweenArguments has been set to 0 so there are no strings allowed between arguments;
- it finds a namedGroupingBracesBrackets called to (see Table 2 on page 45) with argument [in=110,out=-90]
- it finds another namedGroupingBracesBrackets but this time called node with argument [below,align=left,scale=0.5]

Referencing Listing 239 on page 63, we see that the first field in the stringsAllowedBetweenArguments is amalgamate and is set to 1 by default. This is for users who wish to specify their settings in multiple YAML files. For example, by using the settings in either Listing 252 or Listing 253 is equivalent to using the settings in Listing 254.

LISTING 252: amalgamate-demo.yaml

commandCodeBlocks:
  stringsAllowedBetweenArguments:
    - 'more'
    - 'strings'
    - 'here'

LISTING 253: amalgamate-demo1.yaml

commandCodeBlocks:
  stringsAllowedBetweenArguments:
    - amalgamate: 1
    - 'more'
    - 'strings'
    - 'here'

LISTING 254: amalgamate-demo2.yaml

commandCodeBlocks:
  stringsAllowedBetweenArguments:
    - amalgamate: 1
    - 'node'
    - 'at'
    - 'to'
    - 'decoration'
    - '+'
    - '-'
    - 'more'
    - 'strings'
    - 'here'

We specify amalgamate to be set to 0 and in which case any settings loaded prior to those specified, including the default, will be overwritten. For example, using the settings in Listing 255 means that only the strings specified in that field will be used.
5.9 Commands and the strings between their arguments

**Listing 255: amalgamate-demo3.yaml**

```yaml
commandCodeBlocks:
  stringsAllowedBetweenArguments:
  - amalgamate: 0
  - 'further'
  - 'settings'
```

It is important to note that the `amalgamate` field, if used, must be in the first field, and specified using the syntax given in Listings 253 to 255.

We may explore this feature further with the code in Listing 256, whose default output is given in Listing 257.

**Listing 256: for-each.tex**

```latex
\foreach \x/\y in {0/1,1/2}{
  body of foreach
}
```

**Listing 257: for-each default output**

```latex
\foreach \x/\y in {0/1,1/2}{
  body of foreach
}
```

Let’s compare this with the output from using the YAML settings in Listing 259, and running the command

```
cmh:~$ latexindent.pl for-each.tex -l foreach.yaml
```

given in Listing 258.

**Listing 258: for-each.tex using Listing 259**

```latex
\foreach \x/\y in {0/1,1/2}{
  body of foreach
}
```

**Listing 259: foreach.yaml**

```yaml
commandCodeBlocks:
  stringsAllowedBetweenArguments:
  - amalgamate: 0
  - '\x/\y'
  - 'in'
```

You might like to compare the output given in Listing 257 and Listing 258. Note, in particular, in Listing 257 that the `foreach` command has not included any of the subsequent strings, and that the braces have been treated as a namedGroupingBracesBrackets. In Listing 258 the `foreach` command has been allowed to have `\x/\y` and in between arguments because of the settings given in Listing 259.

There are some special command names that do not fit within the names recognised by `latexindent.pl`, the first one of which is `@ifnextchar`. From the perspective of `latexindent.pl`, the whole of the text `@ifnextchar` is a command, because it is immediately followed by sets of mandatory arguments. However, without the `commandNameSpecial` field, `latexindent.pl` would not be able to label it as such, because the `[ is, necessarily, not matched by a closing ]`

For example, consider the sample file in Listing 260, which has default output in Listing 261.

**Listing 260: ifnextchar.tex**

```latex
\parbox{
  @ifnextchar[\{arg 1}{arg 2}
}
```

**Listing 261: ifnextchar.tex default output**

```latex
\parbox{
  @ifnextchar[\{arg 1}{arg 2}
}
```

Notice that in Listing 261 the `parbox` command has been able to indent its body, because `latexindent.pl` has successfully found the command `@ifnextchar` first; the pattern-matching of `latexindent.pl` starts from the `inner most <thing>` and `works outwards`, discussed in more detail on page 111.
For demonstration, we can compare this output with that given in Listing 262 in which the settings from Listing 263 have dictated that no special command names, including the \ifnextchar[] command, should not be searched for specially; as such, the parbox command has been unable to indent its body successfully, because the \ifnextchar[] command has not been found.

\parbox{
  \ifnextchar[{arg 1}{arg 2}
}

The \texttt{amalgamate} field can be used for \texttt{commandNameSpecial}, just as for \texttt{stringsAllowedBetweenArguments}. The same condition holds as stated previously, which we state again here:

\textbf{Warning!}

It is important to note that the \texttt{amalgamate} field, if used, in either \texttt{commandNameSpecial} or \texttt{stringsAllowedBetweenArguments} must be in the first field, and specified using the syntax given in Listings 253 to 255.
The -m (modifylinebreaks) switch

All features described in this section will only be relevant if the -m switch is used.

6.1 Text Wrapping ................................. 70
   6.1.1 Text wrap quick start .................. 70
   6.1.2 textWrapOptions: modifying line breaks by text wrapping .......... 70
   6.1.3 Text wrapping on a per-code-block basis ............................ 73
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs ............ 78
6.3 Combining removeParagraphLineBreaks and textWrapOptions ..................... 84
   6.3.1 text wrapping beforeFindingChildCodeBlocks .......................... 85
6.4 Summary of text wrapping .................................................. 87
6.5 oneSentencePerLine: modifying line breaks for sentences ...................... 88
   6.5.1 sentencesFollow ........................................ 90
   6.5.2 sentencesBeginWith ...................................... 90
   6.5.3 sentencesEndWith ....................................... 91
   6.5.4 Features of the oneSentencePerLine routine ......................... 93
6.5.5 Text wrapping and indenting sentences .................................. 94
6.6 Poly-switches ................................................. 96
   6.6.1 Poly-switches for environments .................................. 96
      6.6.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn
                  Line .................................................. 97
      6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWith
                  LineBreak ............................................. 99
      6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary .... 100
      6.6.1.4 Removing line breaks (poly-switches set to −1) .................. 101
      6.6.1.5 About trailing horizontal space ................................ 102
      6.6.1.6 poly-switch line break removal and blank lines ................ 102
   6.6.2 Poly-switches for double back slash .................................. 104
      6.6.2.1 Double back slash starts on own line .......................... 104
      6.6.2.2 Double back slash finishes with line break ..................... 105
      6.6.2.3 Double back slash poly-switches for specialBeginEnd .......... 105
      6.6.2.4 Double back slash poly-switches for optional and mandatory arguments106
      6.6.2.5 Double back slash optional square brackets .................. 107
   6.6.3 Poly-switches for other code blocks .................................. 107
   6.6.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches .... 109
As of Version 3.0, latexindent.pl has the -m switch, which permits latexindent.pl to modify line breaks, according to the specifications in the modifyLineBreaks field. The settings in this field will only be considered if the -m switch has been used. A snippet of the default settings of this field is shown in Listing 264.

```
LISTING 264: modifyLineBreaks
487 modifyLineBreaks:
488   preserveBlankLines: 1
489   condenseMultipleBlankLinesInto: 1
```

Having read the previous paragraph, it should sound reasonable that, if you call latexindent.pl using the -m switch, then you give it permission to modify line breaks, but let's be clear:

**Warning!**

If you call latexindent.pl with the -m switch, then you are giving it permission to modify line breaks. By default, the only thing that will happen is that multiple blank lines will be condensed into one blank line; many other settings are possible, discussed next.

**preserveBlankLines: 0|1**

This field is directly related to poly-switches, discussed in Section 6.6. By default, it is set to 1, which means that blank lines will be protected from removal; however, regardless of this setting, multiple blank lines can be condensed if condenseMultipleBlankLinesInto is greater than 0, discussed next.

**condenseMultipleBlankLinesInto: (positive integer)**

Assuming that this switch takes an integer value greater than 0, latexindent.pl will condense multiple blank lines into the number of blank lines illustrated by this switch. As an example, Listing 265 shows a sample file with blank lines; upon running

```
cmh:~$ latexindent.pl myfile.tex -m -o=+-mod1
```

the output is shown in Listing 266; note that the multiple blank lines have been condensed into one blank line, and note also that we have used the -m switch!

```
LISTING 265: mlb1.tex
before blank line

after blank line

LISTING 266: mlb1-mod1.tex
before blank line

after blank line

after blank line
```
6.1 Text Wrapping

There are many different configuration options for the text wrapping routine of latexindent.pl, perhaps too many. The following sections are comprehensive, but quite long; in an attempt to be brief, you might begin with the settings given in Section 6.1.1.

6.1.1 Text wrap quick start

Of all the available text wrapping options, I consider Listing 267 to be among the most helpful starting points.

**LISTING 267: textwrap-qs.yaml**

```
modifyLineBreaks:
  textWrapOptions:
    columns: 80  # number of columns
    perCodeBlockBasis: 1  # per-code-block wrap
    beforeFindingChildCodeBlocks: 1  # wrap *before* finding child code blocks
    mainDocument: 1  # apply to main document
    afterHeading: 1  # after headings
    items: 1  # within items
  removeParagraphLineBreaks:
    mainDocument: 1
    afterHeading: 1
    items: 1
    beforeTextWrap: 1  # before wrapping text
```

You can read about `perCodeBlockBasis` in Section 6.1.3 and `removeParagraphLineBreaks` in Section 6.2.

If the settings in Listing 267 do not give your desired output, take a look at the demonstration in Section 6.3.1, in particular Listing 334.

6.1.2 textWrapOptions: modifying line breaks by text wrapping

When the `-m` switch is active latexindent.pl has the ability to wrap text using the options specified in the `textWrapOptions` field, see Listing 268.

**LISTING 268: textWrapOptions**

```
  textWrapOptions:
    columns: 0
```

The value of `columns` specifies the column at which the text should be wrapped.

By default, the value of `columns` is 0, so latexindent.pl will not wrap text; if you change it to a value of 2 or more, then text will be wrapped after the character in the specified column.

By default, the text wrapping routine will operate before the code blocks have been searched for; text wrapping on a `per-code-block` basis is discussed in Section 6.1.3.

We consider the file give in Listing 269 for demonstration.

**LISTING 269: textwrap1.tex**

```
Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.
Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.
```

Using the file `textwrap1.yaml` in Listing 271, and running the command

```
  cmsh:~$ latexindent.pl -m textwrap1.tex -o textwrap1-mod1.tex -l textwrap1.yaml
```

we obtain the output in Listing 270.
6.1 Text Wrapping

Listing 270: textwrap1-mod1.tex

Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

Listing 271: textwrap1.yaml

modifyLineBreaks:
  textWrapOptions:
    columns: 20

The text wrapping routine is performed after verbatim environments have been stored, so verbatim environments and verbatim commands are exempt from the routine. For example, using the file in Listing 272,

Listing 272: textwrap2.tex

Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

\begin{verbatim}
  a long line in a verbatim environment, which will not be broken by latexindent.pl
\end{verbatim}

Here is a verb command: \verb!this will not be text wrapped!

and running the following command and continuing to use textwrap1.yaml from Listing 271,

\texttt{cmh:~$ latexindent.pl -m textwrap2.tex -o textwrap2-mod1.tex -l textwrap1.yaml}

then the output is as in Listing 273.

Listing 273: textwrap2-mod1.tex

Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

\begin{verbatim}
  a long line in a verbatim environment, which will not be broken by latexindent.pl
\end{verbatim}

Here is a verb command: \verb!this will not be text wrapped!

Furthermore, the text wrapping routine is performed after the trailing comments have been stored, and they are also exempt from text wrapping. For example, using the file in Listing 274

Listing 274: textwrap3.tex

Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

Here is a line \texttt{% text wrapping does not apply to comments by latexindent.pl}
and running the following command and continuing to use `textwrap1.yaml` from Listing 271,

```
cmh:~$ latexindent.pl -m textwrap3.tex -o textwrap3-mod1.tex -l textwrap1.yaml
```

then the output is as in Listing 275.

```
Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

Here is a line
% text wrapping does not apply to comments by latexindent.pl
```

The default value of huge is `overflow`, which means that words will *not* be broken by the text wrapping routine, implemented by the `Text::Wrap` [29]. There are options to change the huge option for the `Text::Wrap` module to either `wrap` or `die`. Before modifying the value of huge, please bear in mind the following warning:

---

**Warning!**

Changing the value of huge to anything other than `overflow` will slow down `latexindent.pl` significantly when the `-m` switch is active.

Furthermore, changing huge means that you may have some words or commands(!) split across lines in your `.tex` file, which may affect your output. I do not recommend changing this field.

For example, using the settings in Listings 277 and 279 and running the commands

```
cmh:~$ latexindent.pl -m textwrap4.tex -o=+-mod2A -l textwrap2A.yaml
```

```
cmh:~$ latexindent.pl -m textwrap4.tex -o=+-mod2B -l textwrap2B.yaml
```

gives the respective output in Listings 276 and 278.

```
Listing 267: textwrap4-mod2A.tex
Here is a line of text.
```

```
Listing 277: textwrap2A.yaml
modifyLineBreaks:
   textWrapOptions:
      columns: 3
      huge: wrap
```

```
Listing 267: textwrap4-mod2B.tex
Here is a line of text.
```

```
Listing 278: textwrap2B.yaml
modifyLineBreaks:
   textWrapOptions:
      columns: 3
```
6.1 Text Wrapping

You can also specify the tabstop field as an integer value, which is passed to the text wrap module; see [29] for details. Starting with the code in Listing 280 with settings in Listing 281, and running the command

```
cmh:~$ latexindent.pl -m textwrap-ts.tex -o=+-mod1 -l tabstop.yaml
```

gives the code given in Listing 282.

You can specify separator, break and unexpand options in your settings in analogous ways to those demonstrated in Listings 279 and 281, and they will be passed to the Text::Wrap module. I have not found a useful reason to do this; see [29] for more details.

### 6.1.3 Text wrapping on a per-code-block basis

By default, if the value of columns is greater than 0 and the -m switch is active, then the text wrapping routine will operate before the code blocks have been searched for. This behaviour is customisable; in particular, you can instead instruct latexindent.pl to apply `textWrap` on a per-code-block basis. Thanks to [36] for their help in testing and shaping this feature.

The full details of `textWrapOptions` are shown in Listing 283. In particular, note the field `perCodeBlockBasis: 0`.

The code blocks detailed in Listing 283 are with direct reference to those detailed in Table 2 on page 45.

The only special case is the `mainDocument` field; this is designed for 'chapter'-type files that may contain paragraphs that are not within any other code-blocks. The same notation is used between this feature and the `removeParagraphLineBreaks` described in Listing 302 on page 79; in fact, the two features can even be combined (this is detailed in Section 6.3 on page 84).

Note: `mainDocument` replaces `masterDocument` which was used in previous versions of latexindent.pl. The field `masterDocument` is still supported, but it is anticipated to be removed in a future version, so I recommend using `mainDocument` instead.
6.1 Text Wrapping

Let’s explore these switches with reference to the code given in Listing 284; the text outside of the environment is considered part of the mainDocument.

<table>
<thead>
<tr>
<th>Listing 284: textwrap5.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

```latex
\begin{myenv}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}
```

After the environment; here is a line of text that can be wrapped by latexindent.pl.

With reference to this code block, the settings given in Listings 285 to 287 each give the same output.

<table>
<thead>
<tr>
<th>Listing 285: textwrap3.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 286: textwrap4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>environments: 1</td>
</tr>
<tr>
<td>mainDocument: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 287: textwrap5.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>environments:</td>
</tr>
<tr>
<td>myenv: 1</td>
</tr>
<tr>
<td>mainDocument: 1</td>
</tr>
</tbody>
</table>

Let’s explore the similarities and differences in the equivalent (with respect to Listing 284) syntax specified in Listings 285 to 287:

- in each of Listings 285 to 287 notice that columns: 30;
- in each of Listings 285 to 287 notice that perCodeBlockBasis: 1;
- in Listing 285 we have specified all: 1 so that the text wrapping will operate upon all code blocks;
- in Listing 286 we have not specified all, and instead, have specified that text wrapping should be applied to each of environments and mainDocument;
- in Listing 287 we have specified text wrapping for mainDocument and on a per-name basis for environments code blocks.

Upon running the following commands

```bash
cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap3.yaml -m
cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap4.yaml -m
cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap5.yaml -m
```

we obtain the output shown in Listing 288.

<table>
<thead>
<tr>
<th>Listing 288: textwrap5-mod3.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

```latex
\begin{myenv}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}
```

After the environment; here is a line of text that can be wrapped by latexindent.pl.
We can explore the idea of per-name text wrapping given in Listing 287 by using Listing 289.

<table>
<thead>
<tr>
<th>Listing 289: textwrap6.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>Within the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\begin{another}</td>
</tr>
<tr>
<td>Within the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\end{another}</td>
</tr>
<tr>
<td>After the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

In particular, upon running

```bash
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap5.yaml -m |
```

we obtain the output given in Listing 290.

<table>
<thead>
<tr>
<th>Listing 290: textwrap6.tex using Listing 287</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>Within the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\begin{another}</td>
</tr>
<tr>
<td>Within the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\end{another}</td>
</tr>
<tr>
<td>After the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

Notice that, because environments has been specified only for myenv (in Listing 287) that the environment named another has not had text wrapping applied to it.

The all field can be specified with exceptions which can either be done on a per-code-block or per-name basis; we explore this in relation to Listing 289 in the settings given in Listings 291 to 293.

<table>
<thead>
<tr>
<th>Listing 291: textwrap6.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all:</td>
</tr>
<tr>
<td>except:</td>
</tr>
<tr>
<td>- environments</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 292: textwrap7.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all:</td>
</tr>
<tr>
<td>except:</td>
</tr>
<tr>
<td>- myenv</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 293: textwrap8.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all:</td>
</tr>
<tr>
<td>except:</td>
</tr>
<tr>
<td>- mainDocument</td>
</tr>
</tbody>
</table>

Upon running the commands
we receive the respective output given in Listings 294 to 296.

**Listing 294: textwrap6.tex using Listing 291**

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

**Listing 295: textwrap6.tex using Listing 292**

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

**Listing 296: textwrap6.tex using Listing 293**

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.
Notice that:

- in Listing 294 the text wrapping routine has not been applied to any environments because it has been switched off (per-code-block) in Listing 291;
- in Listing 295 the text wrapping routine has not been applied to myenv because it has been switched off (per-name) in Listing 292;
- in Listing 296 the text wrapping routine has not been applied to mainDocument because of the settings in Listing 293.

The columns field has a variety of different ways that it can be specified; we’ve seen two basic ways already: the default (set to 0) and a positive integer (see Listing 289 on page 75, for example). We explore further options in Listings 297 to 299.

Listing 297 and Listing 298 are equivalent. Upon running the commands

```
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap9.yaml -m
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap11.yaml -m
```

we receive the respective output given in Listings 300 and 301.

Listing 297: textwrap9.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns:
      default: 30
      environments: 50
      perCodeBlockBasis: 1
    all: 1
```

Listing 298: textwrap10.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns:
      default: 30
      environments:
        default: 50
        myenv: 50
        another: 15
        perCodeBlockBasis: 1
    all: 1
```

Listing 299: textwrap11.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns:
      default: 30
      environments:
        myenv: 50
        another: 15
        perCodeBlockBasis: 1
      all: 1
```

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

```
\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}
```

```
\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}
```

After the environment; here is a line of text that can be wrapped by latexindent.pl.
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

Before the environment; here is a line of text that can be wrapped by \texttt{latexindent.pl}.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by \texttt{latexindent.pl}.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by \texttt{latexindent.pl}.
\end{another}

After the environment; here is a line of text that can be wrapped by \texttt{latexindent.pl}.

Notice that:

- in Listing 300 the text for the \texttt{mainDocument} has been wrapped using 30 columns, while environments has been wrapped using 50 columns;
- in Listing 301 the text for \texttt{myenv} has been wrapped using 50 columns, the text for \texttt{another} has been wrapped using 15 columns, and \texttt{mainDocument} has been wrapped using 30 columns.

If you don't specify a default value on per-code-block basis, then the default value from \texttt{columns} will be inherited; if you don't specify a default value for \texttt{columns} then 80 will be used.

\texttt{alignAtAmpersandTakesPriority} is set to 1 by default; assuming that text wrapping is occurring on a per-code-block basis, and the current environment/code block is specified within Listing 39 on page 26 then text wrapping will be disabled for this code block.

If you wish to specify afterHeading commands (see Listing 132 on page 43) on a per-name basis, then you need to append the name with :heading, for example, you might use section:heading.

6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

When the \texttt{-m} switch is active \texttt{latexindent.pl} has the ability to remove line breaks from within paragraphs; the behaviour is controlled by the \texttt{removeParagraphLineBreaks} field, detailed in Listing 302. Thank you to [22] for shaping and assisting with the testing of this feature.

\begin{Verbatim}
\texttt{removeParagraphLineBreaks}: \texttt{\{fields\}}
\end{Verbatim}

This feature is considered complimentary to the oneSentencePerLine feature described in Section 6.5 on page 88.
This routine can be turned on *globally* for every code block type known to `latexindent.pl` (see Table 2 on page 45) by using the `all` switch; by default, this switch is *off*. Assuming that the `all` switch is off, then the routine can be controlled on a per-code-block-type basis, and within that, on a per-name basis. We will consider examples of each of these in turn, but before we do, let’s specify what `latexindent.pl` considers as a paragraph:

- it must begin on its own line with either an alphabetic or numeric character, and not with any of the code-block types detailed in Table 2 on page 45;
- it can include line breaks, but finishes when it meets either a blank line, a `\par` command, or any of the user-specified settings in the `paragraphsStopAt` field, detailed in Listing 319 on page 83.

Let’s start with the `.tex` file in Listing 303, together with the YAML settings in Listing 304.

**Listing 303: shortlines.tex**
```
\begin{myenv}
The lines in this environment are very short and contain many linebreaks.
Another paragraph.
\end{myenv}
```

Upon running the command
```
cmb:~$ latexindent.pl -m shortlines.tex -o shortlines1.tex -l remove-para1.yaml
```
then we obtain the output given in Listing 305.

**Listing 305: shortlines1.tex**
```
\begin{myenv}
   The lines in this environment are very short and contain many linebreaks.
Another paragraph.
\end{myenv}
```

Keen readers may notice that some trailing white space must be present in the file in Listing 303 which has crept in to the output in Listing 305. This can be fixed using the YAML file in Listing 428.
on page 102 and running, for example,

```
cmh:~$ latexindent.pl -m shortlines.tex -o shortlines1-tws.tex -l remove-paral.yaml,removeTWS-before.yaml
```

in which case the output is as in Listing 306; notice that the double spaces present in Listing 305 have been addressed.

```latex
\begin{myenv}
\quad The lines in this environment are very short and contain many linebreaks.
\quad Another paragraph.
\end{myenv}
```

Listing 306: shortlines1-tws.tex

Keeping with the settings in Listing 304, we note that the all switch applies to all code block types. So, for example, let’s consider the files in Listings 307 and 308

```
\begin{myenv}
\mycommand{The lines in this command are very short and contain many linebreaks.}
Another paragraph.
\end{myenv}
```

Listing 307: shortlines-mand.tex

```
\mycommand[The lines in this command are very short and contain many linebreaks.]
Another paragraph.
```

Listing 308: shortlines-opt.tex

Upon running the commands

```
cmh:~$ latexindent.pl -m shortlines-mand.tex -o shortlines-mand1.tex -l remove-paral.yaml
cmh:~$ latexindent.pl -m shortlines-opt.tex -o shortlines-opt1.tex -l remove-paral.yaml
```

then we obtain the respective output given in Listings 309 and 310.

```
\mycommand{
The lines in this command are very short and contain many linebreaks.
Another paragraph.
}
```

Listing 309: shortlines-mand1.tex

```
\mycommand[
The lines in this command are very short and contain many linebreaks.
Another paragraph.
]
```

Listing 310: shortlines-opt1.tex

Assuming that we turn off the all switch (by setting it to 0), then we can control the behaviour of removeParagraphLineBreaks either on a per-code-block-type basis, or on a per-name basis.

For example, let’s use the code in Listing 311, and consider the settings in Listings 312 and 313; note that in Listing 312 we specify that every environment should receive treatment from the routine, while in Listing 313 we specify that only the one environment should receive the treatment.
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

LISTING 311: shortlines-envs.tex
\begin{one}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{one}

\begin{two}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{two}

Upon running the commands

cmh:~$ latexindent.pl -m shortlines-envs.tex -o shortlines-envs2.tex -l remove-para2.yaml
cmh:~$ latexindent.pl -m shortlines-envs.tex -o shortlines-envs3.tex -l remove-para3.yaml

then we obtain the respective output given in Listings 314 and 315.

LISTING 314: shortlines-envs2.tex
\begin{one}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{one}

\begin{two}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{two}
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

The remaining code-block types can be customised in analogous ways, although note that commands, keyEqualsValuesBracesBrackets, namedGroupingBracesBrackets, UnNamedGroupingBracesBrackets are controlled by the optionalArguments and the mandatoryArguments.

The only special case is the mainDocument field; this is designed for 'chapter'-type files that may contain paragraphs that are not within any other code-blocks. For example, consider the file in Listing 316, with the YAML settings in Listing 317.

Note: mainDocument replaces masterDocument which was used in previous versions of latexindent.pl. The field masterDocument is still supported, but it is anticipated to be removed in a future version, so I recommend using mainDocument instead.

Upon running the following command

```
$ latexindent.pl -m shortlines-md.tex -o shortlines-md4.tex -l remove-para4.yaml
```

then we obtain the output in Listing 318.
The lines in this document are very short and contain many linebreaks.

Another paragraph.

\begin{myenv}
The lines in this document are very short and contain many linebreaks.
\end{myenv}

Note that the `all` field can take the same exceptions detailed in Listings 291 to 293.

The `paragraphsStopAt` fields, shown in Listing 319.

The fields specified in `paragraphsStopAt` tell `latexindent.pl` to stop the current paragraph when it reaches a line that begins with any of the code-block types specified as 1 in Listing 319. By default, you'll see that the paragraph line break routine will stop when it reaches an environment or verbatim code block at the beginning of a line. It is not possible to specify these fields on a per-name basis.

Let's use the `.tex` file in Listing 320; we will, in turn, consider the settings in Listings 321 and 322.

Upon using the settings from Listing 317 on the previous page and running the commands
we obtain the respective outputs in Listings 323 to 325; notice in particular that:

- in Listing 323 the paragraph line break routine has included commands and comments;
- in Listing 324 the paragraph line break routine has stopped at the \texttt{emph} command, because in Listing 321 we have specified commands to be 1, and \texttt{emph} is at the beginning of a line;
- in Listing 325 the paragraph line break routine has stopped at the comments, because in Listing 322 we have specified comments to be 1, and the comment is at the beginning of a line.

In all outputs in Listings 323 to 325 we notice that the paragraph line break routine has stopped at \texttt{\begin{myenv}} because, by default, environments is set to 1 in Listing 319 on the preceding page.

6.3 Combining removeParagraphLineBreaks and textWrapOptions

The text wrapping routine (Section 6.1 on page 70) and remove paragraph line breaks routine (Section 6.2 on page 78) can be combined.

We motivate this feature with the code given in Listing 326.
Applying the text wrap routine from Section 6.1 on page 70 with, for example, Listing 285 on page 74 gives the output in Listing 327.

<table>
<thead>
<tr>
<th>LISTING 327: textwrap7.tex using Listing 285</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.</td>
</tr>
</tbody>
</table>

The text wrapping routine has behaved as expected, but it may be desired to remove paragraph line breaks before performing the text wrapping routine. The desired behaviour can be achieved by employing the beforeTextWrap switch.

Explicitly, using the settings in Listing 329 and running the command

```
cmh:~$ latexindent.pl -m textwrap7.tex -l=textwrap12.yaml -o=+-mod12
```

we obtain the output in Listing 328.

<table>
<thead>
<tr>
<th>LISTING 328: textwrap7-mod12.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.</td>
</tr>
</tbody>
</table>

In Listing 328 the paragraph line breaks have first been removed from Listing 326, and then the text wrapping routine has been applied. It is envisaged that variants of Listing 329 will be among the most useful settings for these two features.

### 6.3.1 text wrapping beforeFindingChildCodeBlocks

I think it likely that most users will wish to employ the beforeFindingChildCodeBlocks option for the text wrap routine.

To motivate its use, we begin with the file in Listing 330.

<table>
<thead>
<tr>
<th>LISTING 330: textwrap-bfccb.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>one \test{two three four five six seven eight nine} ten eleven twelve thirteen fourteen fifteen sixteen seventeen</td>
</tr>
</tbody>
</table>

Using the settings in Listing 329 and running

```
cmh:~$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap12.yaml -o=+-mod12
```

gives the output in Listing 331
6.3 Combining removeParagraphLineBreaks and textWrapOptions

---

**LISTING 331: textwrap-bfccb-mod12.tex**

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
<tbody>
<tr>
<td>\test{test five six seven eight nine}</td>
<td>ten</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eleven twelve thirteen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fourteen fifteen sixteen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>seventeen</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that we have added a ‘ruler’ to **Listing 331** to assist with measuring.

The output in **Listing 331** is not ideal, but it is expected. The reasoning is as follows:

- **latexindent.pl** first of all searches for code blocks (see Table 2 on page 45);
- it replaces each code block with a unique identifying string;
- with the settings of **Listing 329** in place, it performs the paragraph line break removal, and then the text wrapping routine first of all on the **text** command, and then on the surrounding text;
- the surrounding text does not know that **text** is a command.

We can instruct **latexindent.pl** to perform text wrapping before searching for child code blocks by using the **beforeFindingChildCodeBlocks** field.

We save the quick-start settings from **Listing 267** into **Listing 332** and change the value of columns for demonstration. Upon running the command:

```
$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap13.yaml -o=+-mod13
```

we receive the output in **Listing 333**.

---

**LISTING 332: textwrap13.yaml (tweaked quick start)**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 40 #<--- Changed from quick start
    perCodeBlockBasis: 1
    beforeFindingChildCodeBlocks: 1
    mainDocument: 1
    afterHeading: 1
    items: 1
  removeParagraphLineBreaks:
    mainDocument: 1
    afterHeading: 1
    items: 1
    beforeTextWrap: 1
```

---

**LISTING 333: textwrap-bfccb-mod13.tex**

<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
</tr>
</thead>
<tbody>
<tr>
<td>\test{test five six seven eight nine}</td>
<td>ten eleven twelve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thirteen fourteen fifteen sixteen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>seventeen</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This output is different from **Listing 331**, but is still not ideal, as the **test** command has indented its mandatory argument. We can employ **noAdditionalIndent** from Section 5.8 on page 46 in **Listing 335** and run the command.
6.4 Summary of text wrapping

I consider the most useful starting point for text wrapping to be given in Section 6.1.1 and Section 6.3.1.

Starting from Listing 267, it is likely that you will have to experiment with making adjustments (such as that given in Listing 335) depending on your preference.

It is important to note the following:

- verbatim code blocks of all types will not be affected by the text wrapping routine. See the demonstration in Listing 273 on page 71, together with environments: Listing 18 on page 22, commands: Listing 19 on page 22, noIndentBlock: Listing 24, specialBeginEnd: Listing 126 on page 41;
- comments will not be affected by the text wrapping routine (see Listing 275 on page 72);
- it is possible to wrap text on a per-code-block and a per-name basis;

For reference, let’s say that we had started from Listing 329, which instructs latexindent.pl to apply the text-wrapping and paragraph-line-break-removal routines to all code blocks. In order to achieve the output in Listing 334, then we would need to employ an exception, which we demonstrate in Listing 336.
• indentation is performed after the text wrapping routine; as such, indented code will likely exceed any maximum value set in the columns field.

6.5 **oneSentencePerLine: modifying line breaks for sentences**

You can instruct `latexindent.pl` to format your file so that it puts one sentence per line. Thank you to [20] for helping to shape and test this feature. The behaviour of this part of the script is controlled by the switches detailed in Listing 337, all of which we discuss next.

### Listing 337: oneSentencePerLine

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>490</td>
<td><code>oneSentencePerLine:</code></td>
</tr>
<tr>
<td>491</td>
<td><code>manipulateSentences: 0</code></td>
</tr>
<tr>
<td>492</td>
<td><code>removeSentenceLineBreaks: 1</code></td>
</tr>
<tr>
<td>493</td>
<td><code>textWrapSentences: 0</code></td>
</tr>
<tr>
<td>494</td>
<td><code>sentenceIndent: &quot;&quot;</code></td>
</tr>
<tr>
<td>495</td>
<td><code>sentencesFollow:</code></td>
</tr>
<tr>
<td>496</td>
<td><code>par: 1</code></td>
</tr>
<tr>
<td>497</td>
<td><code>blankLine: 1</code></td>
</tr>
<tr>
<td>498</td>
<td><code>fullStop: 1</code></td>
</tr>
<tr>
<td>499</td>
<td><code>exclamationMark: 1</code></td>
</tr>
<tr>
<td>500</td>
<td><code>questionMark: 1</code></td>
</tr>
<tr>
<td>501</td>
<td><code>rightBrace: 1</code></td>
</tr>
<tr>
<td>502</td>
<td><code>commentOnPreviousLine: 1</code></td>
</tr>
<tr>
<td>503</td>
<td><code>other: 0</code></td>
</tr>
<tr>
<td>504</td>
<td><code>sentencesBeginWith:</code></td>
</tr>
<tr>
<td>505</td>
<td><code>A-Z: 1</code></td>
</tr>
<tr>
<td>506</td>
<td><code>a-z: 0</code></td>
</tr>
<tr>
<td>507</td>
<td><code>other: 0</code></td>
</tr>
<tr>
<td>508</td>
<td><code>sentencesEndWith:</code></td>
</tr>
<tr>
<td>509</td>
<td><code>basicFullStop: 0</code></td>
</tr>
<tr>
<td>510</td>
<td><code>betterFullStop: 1</code></td>
</tr>
<tr>
<td>511</td>
<td><code>exclamationMark: 1</code></td>
</tr>
<tr>
<td>512</td>
<td><code>questionMark: 1</code></td>
</tr>
<tr>
<td>513</td>
<td><code>other: 0</code></td>
</tr>
</tbody>
</table>

**manipulateSentences**: 0|1

This is a binary switch that details if `latexindent.pl` should perform the sentence manipulation routine; it is off (set to 0) by default, and you will need to turn it on (by setting it to 1) if you want the script to modify line breaks surrounding and within sentences.

**removeSentenceLineBreaks**: 0|1

When operating upon sentences `latexindent.pl` will, by default, remove internal line breaks as `removeSentenceLineBreaks` is set to 1. Setting this switch to 0 instructs `latexindent.pl` not to do so.

For example, consider `multiple-sentences.tex` shown in Listing 338.

### Listing 338: multiple-sentences.tex

This is the first sentence. This is the; second, sentence. This is the third sentence.

This is the fourth sentence! This is the fifth sentence? This is the sixth sentence.

If we use the YAML files in Listings 340 and 342, and run the commands...
then we obtain the respective output given in Listings 339 and 341.

**Listing 339: multiple-sentences.tex using Listing 340**

This is the first sentence.
This is the; second, sentence.
This is the third sentence.

This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

**Listing 341: multiple-sentences.tex using Listing 342**

This is the first sentence.
This is the; second, sentence.
This is the third sentence.

This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

Notice, in particular, that the ‘internal’ sentence line breaks in Listing 338 have been removed in Listing 339, but have not been removed in Listing 341.

The remainder of the settings displayed in Listing 337 on the preceding page instruct `latexindent.pl` on how to define a sentence. From the perspective of `latexindent.pl` a sentence must:

- **follow** a certain character or set of characters (see Listing 343); by default, this is either `\par`, a blank line, a full stop/period (`.`), exclamation mark (`!`), question mark (`?`) right brace (`}`) or a comment on the previous line;
- **begin** with a character type (see Listing 344); by default, this is only capital letters;
- **end** with a character (see Listing 345); by default, these are full stop/period (`.`), exclamation mark (`!`) and question mark (`?`).

In each case, you can specify the other field to include any pattern that you would like; you can specify anything in this field using the language of regular expressions.
6.5 oneSentencePerLine: modifying line breaks for sentences

6.5.1 sentencesFollow

Let's explore a few of the switches in sentencesFollow; let's start with Listing 338 on page 88, and use the YAML settings given in Listing 347. Using the command

```
cmh:~$ latexindent.pl multiple-sentences -m -l=sentences-follow1.yaml
```

we obtain the output given in Listing 346.

<table>
<thead>
<tr>
<th>Listing 346: multiple-sentences.tex using Listing 347</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the first sentence.</td>
</tr>
<tr>
<td>This is the second sentence.</td>
</tr>
<tr>
<td>This is the third sentence.</td>
</tr>
</tbody>
</table>

This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

Notice that, because blankLine is set to 0, latexindent.pl will not seek sentences following a blank line, and so the fourth sentence has not been accounted for.

We can explore the other field in Listing 343 with the .tex file detailed in Listing 348.

<table>
<thead>
<tr>
<th>Listing 347: sentences-follow1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine:</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
<tr>
<td>sentencesFollow:</td>
</tr>
<tr>
<td>blankLine: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 348: multiple-sentences1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Some sentences stand alone in brackets.) This is the first sentence. This is the second sentence. This is the third sentence.</td>
</tr>
</tbody>
</table>

Upon running the following commands

```
cmh:~$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml

cmh:~$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml,sentences-follow2.yaml
```

then we obtain the respective output given in Listings 349 and 350.

<table>
<thead>
<tr>
<th>Listing 349: multiple-sentences1.tex using Listing 340 on the preceding page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Some sentences stand alone in brackets.) This is the first sentence. This is the second sentence. This is the third sentence.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 350: multiple-sentences1.tex using Listing 351</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Some sentences stand alone in brackets.) This is the first sentence. This is the second sentence. This is the third sentence.</td>
</tr>
</tbody>
</table>

Notice that in Listing 349 the first sentence after the ) has not been accounted for, but that following the inclusion of Listing 351, the output given in Listing 350 demonstrates that the sentence has been accounted for correctly.

6.5.2 sentencesBeginWith

By default, latexindent.pl will only assume that sentences begin with the upper case letters A-Z; you can instruct the script to define sentences to begin with lower case letters (see Listing 344), and we can use the other field to define sentences to begin with other characters.
This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.

Upon running the following commands

```
cmh:~$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml
```

then we obtain the respective output given in Listings 353 and 354.

This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.

Notice that in Listing 353, the first sentence has been accounted for but that the subsequent sentences have not. In Listing 354, all of the sentences have been accounted for, because the other field in Listing 355 has defined sentences to begin with either \$ or any numeric digit, 0 to 9.

### 6.5.3 sentencesEndWith

Let's return to Listing 338 on page 88; we have already seen the default way in which `latexindent.pl` will operate on the sentences in this file in Listing 339 on page 89. We can populate the other field with any character that we wish; for example, using the YAML specified in Listing 357 and the command

```
$ latexindent.pl multiple-sentences -m -l=sentences-end1.yaml
```

then we obtain the output in Listing 356.

This is the first sentence.
This is the second sentence.
This is the third sentence.
This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.
6.5 oneSentencePerLine: modifying line breaks for sentences

There is a subtle difference between the output in Listings 356 and 358; in particular, in Listing 356 the word sentence has not been defined as a sentence, because we have not instructed latexindent.pl to begin sentences with lower case letters. We have changed this by using the settings in Listing 359, and the associated output in Listing 358 reflects this.

Referencing Listing 345 on page 89, you’ll notice that there is a field called basicFullStop, which is set to 0, and that the betterFullStop is set to 1 by default.

Let’s consider the file shown in Listing 360.

Upon running the following commands

```cmh
$ latexindent.pl url -m -l=manipulate-sentences.yaml
```

we obtain the output given in Listing 361.

Notice that the full stop within the url has been interpreted correctly. This is because, within the betterFullStop, full stops at the end of sentences have the following properties:

- they are ignored within e.g. and i.e.;
- they can not be immediately followed by a lower case or upper case letter;
- they can not be immediately followed by a hyphen, comma, or number.

If you find that the betterFullStop does not work for your purposes, then you can switch it off by setting it to 0, and you can experiment with the other field. You can also seek to customise the betterFullStop routine by using the fine tuning, detailed in Listing 521 on page 128.

The basicFullStop routine should probably be avoided in most situations, as it does not accommodate the specifications above. For example, using the following command

```cmh
$ latexindent.pl url -m -l=alt-full-stop1.yaml
```

and the YAML in Listing 363 gives the output in Listing 362.
6.5 oneSentencePerLine: modifying line breaks for sentences

<table>
<thead>
<tr>
<th>LISTING 362: url.tex using Listing 363</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence, \url{tex.stackexchange.com/} finishes here. Second sentence.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 363: alt-full-stop1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine:</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
<tr>
<td>sentencesEndWith:</td>
</tr>
<tr>
<td>basicFullStop: 1</td>
</tr>
<tr>
<td>betterFullStop: 0</td>
</tr>
</tbody>
</table>

Notice that the full stop within the URL has not been accommodated correctly because of the non-default settings in Listing 363.

6.5.4 Features of the oneSentencePerLine routine

The sentence manipulation routine takes place after verbatim environments, preamble and trailing comments have been accounted for; this means that any characters within these types of code blocks will not be part of the sentence manipulation routine.

For example, if we begin with the .tex file in Listing 364, and run the command

```bash
cmh:~$ latexindent.pl multiple-sentences3 -m -l=manipulate-sentences.yaml
```
then we obtain the output in Listing 365.

<table>
<thead>
<tr>
<th>LISTING 364: multiple-sentences3.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first sentence continues after the verbatim \begin{verbatim}</td>
</tr>
<tr>
<td>\begin{verbatim}</td>
</tr>
<tr>
<td>\end{verbatim}</td>
</tr>
<tr>
<td>\begin{verbatim}</td>
</tr>
<tr>
<td>\end{verbatim}</td>
</tr>
<tr>
<td>contains trailing comments,</td>
</tr>
<tr>
<td>which are ignored.</td>
</tr>
</tbody>
</table>

Furthermore, if sentences run across environments then, by default, the line breaks internal to the sentence will be removed. For example, if we use the .tex file in Listing 366 and run the commands

```bash
cmh:~$ latexindent.pl multiple-sentences4 -m -l=manipulate-sentences.yaml
cmh:~$ latexindent.pl multiple-sentences4 -m -l=keep-sen-line-breaks.yaml
```
then we obtain the output in Listings 367 and 368.
6.5  oneSentencePerLine: modifying line breaks for sentences

Once you've read Section 6.6, you will know that you can accommodate the removal of internal sentence line breaks by using the YAML in Listing 370 and the command:

```bash
$ latexindent -m -l=item-rules2.yaml
```
the output of which is shown in Listing 369.

6.5.5  Text wrapping and indenting sentences

The oneSentencePerLine can be instructed to perform text wrapping and indentation upon sentences.

Let's use the code in Listing 371.

Referencing Listing 373, and running the following command:

```bash
$ latexindent -m -l=sentence-wrap1.yaml
```
we receive the output given in Listing 372.

### Listing 372: multiple-sentences5.tex using Listing 373

A distinção entre conteúdo \textit{real} e conteúdo \textit{intencional} está relacionada, ainda, a distinção entre o conceito husserliano de \textit{experiência} e o uso popular desse termo. No sentido comum, o \textsc{term}{experimentado} e um complexo de eventos exteriores, e o \textsc{term}{experimentar} consiste em percepções (alem de julgamentos e outros atos) nas quais tais eventos aparecem como objetos, e objetos frequentemente ao end.

If you wish to specify the columns field on a per-code-block basis for sentences, then you would use sentence; explicitly, starting with Listing 297 on page 77, for example, you would replace/append environments with, for example, sentence: 50.

If you specify textWrapSentences as 1, but do not specify a value for columns then the text wrapping will not operate on sentences, and you will see a warning in indent.log.

The indentation of sentences requires that sentences are stored as code blocks. This means that you may need to tweak Listing 345 on page 89. Let's explore this in relation to Listing 374.

### Listing 374: multiple-sentences6.tex

Consider the following:

\begin{itemize}
    \item firstly.
    \item secondly.
\end{itemize}

By default, latexindent.pl will find the full-stop within the first \texttt{item}, which means that, upon running the following commands

\begin{verbatim}
$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
    -y="modifyLineBreaks:oneSentencePerLine:sentenceIndent:''"
\end{verbatim}

we receive the respective output in Listing 375 and Listing 376.

### Listing 375: multiple-sentences6-mod1.tex using Listing 373

Consider the following: \begin{itemize} \item firstly. \item secondly. \end{itemize}

We note that Listing 375 the \texttt{itemize} code block has not been indented appropriately. This is because the oneSentencePerLine has been instructed to store sentences (because Listing 373); each sentence is then searched for code blocks.

We can tweak the settings in Listing 345 on page 89 to ensure that full stops are not followed by \texttt{item} commands, and that the end of sentences contains \texttt{\end{itemize}} as in Listing 377 (if you intend to use this, ensure that you remove the line breaks from the other field).
Upon running

```bash
cmh:~$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml,itemize.yaml
```

we receive the output in Listing 378.

Consider the following:

```latex
\begin{itemize}
  \item firstly.
  \item secondly.
\end{itemize}
```

Notice that the sentence has received indentation, and that the `itemize` code block has been found and indented correctly.

### 6.6 Poly-switches

Every other field in the `modifyLineBreaks` field uses poly-switches, and can take one of the following integer values:

- 1 **remove mode**: line breaks before or after the `<part of thing>` can be removed (assuming that `preserveBlankLines` is set to 0);

0 **off mode**: line breaks will not be modified for the `<part of thing>` under consideration;

1 **add mode**: a line break will be added before or after the `<part of thing>` under consideration, assuming that there is not already a line break before or after the `<part of thing>`;

2 **comment then add mode**: a comment symbol will be added, followed by a line break before or after the `<part of thing>` under consideration, assuming that there is not already a comment and line break before or after the `<part of thing>`;

3 **add then blank line mode**: a line break will be added before or after the `<part of thing>` under consideration, assuming that there is not already a line break before or after the `<part of thing>`, followed by a blank line;

4 **add blank line mode**: a blank line will be added before or after the `<part of thing>` under consideration, even if the `<part of thing>` is already on its own line.

In the above, `<part of thing>` refers to either the `begin statement`, `body` or `end statement` of the code blocks detailed in Table 2 on page 45. All poly-switches are `off` by default; `latexindent.pl` searches first of all for per-name settings, and then followed by global per-thing settings.

#### 6.6.1 Poly-switches for environments

We start by viewing a snippet of `defaultSettings.yaml` in Listing 379; note that it contains `global` settings (immediately after the `environments` field) and that `per-name` settings are also allowed – in the case of Listing 379, settings for `equation*` have been specified for demonstration. Note that all poly-switches are `off` (set to 0) by default.
Let’s begin with the simple example given in Listing 380; note that we have annotated key parts of the file using ♠, ♥, ♦ and ♣, these will be related to fields specified in Listing 379.

Let’s now change each of the 1 values in Listings 381 and 382 so that they are 2 and save them into env-mlb3.yaml and env-mlb4.yaml respectively (see Listings 385 and 386).

Upon running commands analogous to the above, we obtain Listings 387 and 388.
Note that line breaks have been added as in Listings 383 and 384, but this time a comment symbol has been added before adding the line break; in both cases, trailing horizontal space has been stripped before doing so.

Let’s now change each of the 1 values in Listings 381 and 382 so that they are 3 and save them into env-mlb5.yaml and env-mlb6.yaml respectively (see Listings 389 and 390).

Upon running commands analogous to the above, we obtain Listings 391 and 392.

Let’s now change each of the 1 values in Listings 389 and 390 so that they are 4 and save them into env-beg4.yaml and env-body4.yaml respectively (see Listings 393 and 394).

We will demonstrate this poly-switch value using the code in Listing 395.

Upon running the commands

```
cmh:∼$ latexindent.pl -m env-mlb1.tex -l env-beg4.yaml
cmh:∼$ latexindent.pl -m env-mlb1.tex -l env-body4.yaml
```
then we receive the respective outputs in Listings 396 and 397.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 396 a blank line has been inserted before the `\begin` statement, even though the `\begin` statement was already on its own line;
2. in Listing 397 a blank line has been inserted before the beginning of the body, even though it already began on its own line.

### 6.6.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWithLineBreak

Let's explore EndStartsOnOwnLine and EndFinishesWithLineBreak in Listings 398 and 399, and in particular, let's allow each of them in turn to take a value of 1.

#### LISTING 398: env-mlb7.yaml
```
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 1
```

#### LISTING 399: env-mlb8.yaml
```
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 1
```

After running the following commands,
```
cmh:$ latexindent.pl -m env-mlb.tex -l env-mlb7.yaml
cmh:$ latexindent.pl -m env-mlb.tex -l env-mlb8.yaml
```
the output is as in Listings 400 and 401.

#### LISTING 400: env-mlb.tex using Listing 398
```
before words \begin{myenv}body of myenv after words
```

#### LISTING 401: env-mlb.tex using Listing 399
```
before words \begin{myenv}body of myenv\end{myenv} after words
```

There are a couple of points to note:
- in Listing 400 a line break has been added at the point denoted by ♦ in Listing 380 on page 97; no other line breaks have been changed and the \end{myenv} statement has not received indentation (as intended);
- in Listing 401 a line break has been added at the point denoted by ♣ in Listing 380 on page 97.

Let's now change each of the 1 values in Listings 398 and 399 so that they are 2 and save them into env-mlb9.yaml and env-mlb10.yaml respectively (see Listings 402 and 403).

#### LISTING 402: env-mlb9.yaml
```
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 2
```

#### LISTING 403: env-mlb10.yaml
```
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 2
```

Upon running commands analogous to the above, we obtain Listings 404 and 405.

#### LISTING 404: env-mlb.tex using Listing 402
```
before words \begin{myenv}body of myenv\% after words
```

#### LISTING 405: env-mlb.tex using Listing 403
```
before words \begin{myenv}body of myenv\end{myenv}\% after words
```

Note that line breaks have been added as in Listings 400 and 401, but this time a comment symbol has been added before adding the line break; in both cases, trailing horizontal space has been stripped before doing so.

Let's now change each of the 1 values in Listings 398 and 399 so that they are 3 and save them into env-mlb11.yaml and env-mlb12.yaml respectively (see Listings 406 and 407).

#### LISTING 406: env-mlb11.yaml
```
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 3
```

#### LISTING 407: env-mlb12.yaml
```
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 3
```

Upon running commands analogous to the above, we obtain Listings 408 and 409.
Note that line breaks have been added as in Listings 400 and 401, and that a blank line has been added after the line break.

Let’s now change each of the 1 values in Listings 406 and 407 so that they are 4 and save them into env-end4.yaml and env-end-f4.yaml respectively (see Listings 410 and 411).

We will demonstrate this poly-switch value using the code from Listing 395 on page 98.

Upon running the commands

```
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-end4.yaml
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-end-f4.yaml
```

then we receive the respective outputs in Listings 412 and 413.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 412 a blank line has been inserted before the \end statement, even though the \end statement was already on its own line;
2. in Listing 413 a blank line has been inserted after the \end statement, even though it already began on its own line.

### 6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary

If you ask latexindent.pl to add a line break (possibly with a comment) using a poly-switch value of 1 (or 2 or 3), it will only do so if necessary. For example, if you process the file in Listing 414 using poly-switch values of 1, 2, or 3, it will be left unchanged.

Setting the poly-switches to a value of 4 instructs latexindent.pl to add a line break even if the <part of thing> is already on its own line; see Listings 396 and 397 and Listings 412 and 413.

In contrast, the output from processing the file in Listing 415 will vary depending on the poly-switches used; in Listing 416 you’ll see that the comment symbol after the \begin{myenv} has been moved to the next line, as BodyStartsOnOwnLine is set to 1. In Listing 417 you’ll see that the comment has been accounted for correctly because BodyStartsOnOwnLine has been set to 2, and
the comment symbol has not been moved to its own line. You’re encouraged to experiment with Listing 415 and by setting the other poly-switches considered so far to 2 in turn.

6.6.1.4 Removing line breaks (poly-switches set to −1)
Setting poly-switches to −1 tells latexindent.pl to remove line breaks of the \<part of the thing\>, if necessary. We will consider the example code given in Listing 418, noting in particular the positions of the line break highlighters, ♠, ♥, ♦ and ♣, together with the associated YAML files in Listings 419 to 422.

After running the commands

```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb13.yaml
```
```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb14.yaml
```
```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb15.yaml
```
```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb16.yaml
```
we obtain the respective output in Listings 423 to 426.
6.6 Poly-switches

### 6.6.1.5 About trailing horizontal space

Recall that on page 26 we discussed the YAML field `removeTrailingWhitespace`, and that it has two (binary) switches to determine if horizontal space should be removed before processing and after processing. The `beforeProcessing` switch is particularly relevant when considering the `-m` switch; let’s consider the file shown in Listing 427, which highlights trailing spaces.

**Listing 427:** `env-mlb5.tex`
```
\begin{myenv}
body of myenv\end{myenv}
```

The output from the following commands
```
cmh:~$ latexindent.pl -m env-mlb5.tex -l env-mlb13.yaml,env-mlb14.yaml,env-mlb15.yaml,env-mlb16.yaml,removeTWS-before.yaml
```

is shown, respectively, in Listings 429 and 430; note that the trailing horizontal white space has been preserved (by default) in Listing 429, while in Listing 430, it has been removed using the switch specified in Listing 428.

**Listing 429:** `env-mlb5.tex` using Listings 423 to 426
```
\begin{myenv}body of myenv\end{myenv}
```

**Listing 430:** `env-mlb5.tex` using Listings 423 to 426 and Listing 428
```
\begin{myenv}body of myenv\end{myenv}
```

### 6.6.1.6 poly-switch line break removal and blank lines

Now let’s consider the file in Listing 431, which contains blank lines.
Upon running the following commands


we receive the respective outputs in Listings 433 and 434. In Listing 433 we see that the multiple blank lines have each been condensed into one blank line, but that blank lines have not been removed by the poly-switches – this is because, by default, preserveBlankLines is set to 1. By contrast, in Listing 434, we have allowed the poly-switches to remove blank lines because, in Listing 432, we have set preserveBlankLines to 0.

We can explore this further using the blank-line poly-switch value of 3; let's use the file given in Listing 435.

Upon running the following commands

cmh:\$ latexindent.pl -m env-mlb7.tex -l env-mlb13.yaml,env-mlb14.yaml

cmh:\$ latexindent.pl -m env-mlb7.tex -l env-mlb13.yaml,env-mlb14.yaml,UnpreserveBlankLines.yaml

we receive the outputs given in Listings 436 and 437.
6.6 Poly-switches

6.6.2 Poly-switches for double back slash

With reference to `lookForAlignDelims` (see Listing 39 on page 26) you can specify poly-switches to dictate the line-break behaviour of double back slashes in environments (Listing 41 on page 27), commands (Listing 75 on page 33), or special code blocks (Listing 114 on page 39). Note that for these poly-switches to take effect, the name of the code block must necessarily be specified within `lookForAlignDelims` (Listing 39 on page 26); we will demonstrate this in what follows.

Consider the code given in Listing 438.

Referencing Listing 438:

- DBS stands for `double back slash`;
- line breaks ahead of the double back slash are annotated by `★`, and are controlled by `DBSStartsOnOwnLine`;
- line breaks after the double back slash are annotated by `□`, and are controlled by `DBSFinishesWithLineBreak`.

Let's explore each of these in turn.

6.6.2.1 Double back slash starts on own line

We explore `DBSStartsOnOwnLine` (`★` in Listing 438); starting with the code in Listing 438, together with the YAML files given in Listing 440 and Listing 442 and running the following commands

```
cmh:~$ latexindent.pl -m tabular3.tex -l DBS1.yaml
```

then we receive the respective output given in Listing 439 and Listing 441.

<table>
<thead>
<tr>
<th>Listing 439: tabular3.tex using Listing 440</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{tabular}{cc}</code></td>
</tr>
<tr>
<td>1 &amp; 2</td>
</tr>
<tr>
<td><code>\\</code> 3 &amp; 4</td>
</tr>
<tr>
<td><code>\</code></td>
</tr>
<tr>
<td><code>{end{tabular}</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 440: DBS1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>modifyLineBreaks:</code></td>
</tr>
<tr>
<td><code>environments:</code></td>
</tr>
<tr>
<td><code>DBSStartsOnOwnLine:</code> 1</td>
</tr>
</tbody>
</table>

Notice that in:

- Listing 436 that `\end{one}` has added a blank line, because of the value of `EndFinishesWithLineBreak` in Listing 407 on page 99, and even though the line break ahead of `\begin{two}` should have been removed (because of `BeginStartsOnOwnLine` in Listing 419 on page 101), the blank line has been preserved by default;
- Listing 437, by contrast, has had the additional line-break removed, because of the settings in Listing 432.
6.6 Poly-switches

6.6.2 Double back slash finishes with line break

Let’s now explore DBSFinishesWithLineBreak (□ in Listing 438); starting with the code in Listing 438, together with the YAML files given in Listing 444 and Listing 446 and running the following commands

```
cmh:~$ latexindent -m tabular3.tex -l DBS3.yaml
cmh:~$ latexindent -m tabular3.tex -l DBS4.yaml
```

then we receive the respective output given in Listing 443 and Listing 445.

We note that

- Listing 444 specifies DBSFinishesWithLineBreak for every environment (that is within `lookForAlignDelims`, Listing 42 on page 27); the code following the double back slashes from Listing 438 has been moved to their own line in Listing 443;
- Listing 446 specifies DBSFinishesWithLineBreak on a per-name basis for `tabular` (that is within `lookForAlignDelims`, Listing 42 on page 27); the first double back slashes from Listing 438 have moved code following them to their own line in Listing 445, having added comment symbols before moving them; the final double back slashes have not added a line break as they are at the end of the body within the code block.

6.6.2.3 Double back slash poly-switches for specialBeginEnd

Let’s explore the double back slash poly-switches for code blocks within `specialBeginEnd` code blocks (Listing 112 on page 39); we begin with the code within Listing 447.
6.6 Poly-switches

Upon using the YAML settings in Listing 449, and running the command

```
$ latexindent
.pl -m special4.tex -l DBS5.yaml
```

then we receive the output given in Listing 448.

There are a few things to note:

- in Listing 449 we have specified `cmhMath` within `lookForAlignDelims`; without this, the double back slash poly-switches would be ignored for this code block;
- the `DBSFinishesWithLineBreak` poly-switch has controlled the line breaks following the double back slashes;
- the `SpecialEndStartsOnOwnLine` poly-switch has controlled the addition of a comment symbol, followed by a line break, as it is set to a value of 2.

### 6.6.2.4 Double back slash poly-switches for optional and mandatory arguments

For clarity, we provide a demonstration of controlling the double back slash poly-switches for optional and mandatory arguments. We begin with the code in Listing 450.

```
\mycommand [1&2 \ &3 [4&5&6] [7&8 \ &9 [10&11&12]]
```

Upon using the YAML settings in Listings 452 and 454, and running the command

```
$ latexindent
.pl -m mycommand2.tex -l DBS6.yaml
$ latexindent
.pl -m mycommand2.tex -l DBS7.yaml
```

then we receive the output given in Listings 451 and 453.
6.6.2.5 Double back slash optional square brackets

The pattern matching for the double back slash will also, optionally, allow trailing square brackets that contain a measurement of vertical spacing, for example `\[3pt]`.

For example, beginning with the code in Listing 455

```
\begin{pmatrix}
1 & 2 \\
3 & 4 \\
5 & 6 \\
7 & 8
\end{pmatrix}
```

and running the following command, using Listing 444,

```
cmh:~$ latexindent.pl -m pmatrix3.tex -l DBS3.yaml
```

then we receive the output given in Listing 456.

```
\begin{pmatrix}
1 & 2 \\
3 & 4 \\
5 & 6 \\
7 & 8
\end{pmatrix}
```

You can customise the pattern for the double back slash by exploring the `fine tuning` field detailed in Listing 521 on page 128.

6.6.3 Poly-switches for other code blocks

Rather than repeat the examples shown for the environment code blocks (in Section 6.6.1 on page 96), we choose to detail the poly-switches for all other code blocks in Table 3; note that each and every one of these poly-switches is `off by default`, i.e. set to 0.

Note also that, by design, line breaks involving, `filecontents` and `comment-marked` code blocks (Listing 76 on page 33) can not be modified using `latexindent.pl`. However, there are two poly-switches available for `verbatim` code blocks: `environments` (Listing 18 on page 22), commands (Listing 19 on page 22) and `specialBeginEnd` (Listing 125 on page 41).
### Table 3: Poly-switch mappings for all code-block types

<table>
<thead>
<tr>
<th>Code block</th>
<th>Sample</th>
<th>Poly-switch mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>environment</td>
<td>before words♠</td>
<td>♠ BeginStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\begin{myenv}♥</td>
<td>♥ BodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>body of myenv♦</td>
<td>♦ EndStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\end{myenv}♠</td>
<td>♣ EndFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>after words</td>
<td></td>
</tr>
<tr>
<td>ifelsefi</td>
<td>before words♠</td>
<td>♠ IfStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\if...♥</td>
<td>♥ BodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>body of if/or statement▲</td>
<td>▲ OrStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\or♥</td>
<td>♥ OrFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>body of if/or statement★</td>
<td>★ ElseStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\else□</td>
<td>□ ElseFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>body of else statement♦</td>
<td>◆ FiStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\fi♣</td>
<td>♣ FiFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>after words</td>
<td></td>
</tr>
<tr>
<td>optionalArguments</td>
<td>...♠</td>
<td>♠ LSqBStartsOnOwnLine⁸</td>
</tr>
<tr>
<td></td>
<td>[♥ value before comma★, □ end of body of opt arg♦ ]♠</td>
<td>♥ OptArgBodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>□ CommaStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>◆ CommaFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td></td>
<td>♣ RSqBStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>♣ RSqBFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>...♠</td>
<td>♠ LCuBStartsOnOwnLine⁹</td>
</tr>
<tr>
<td></td>
<td>{♥ value before comma★, □ end of body of mand arg♦ }♠</td>
<td>♥ MandArgBodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>□ CommaStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>□ CommaFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td></td>
<td>◆ RCuBStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>♣ RCuBFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>commands</td>
<td>before words♠</td>
<td>♠ CommandStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\mycommand♥</td>
<td>♥ CommandNameFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>(arguments)</td>
<td></td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>before words♠</td>
<td>♠ NameStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>myname♥</td>
<td>♥ NameFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>(braces/brackets)</td>
<td></td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>before words♠</td>
<td>♠ KeyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>key★=♥</td>
<td>♥ EqualsStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>(braces/brackets)</td>
<td>♥ EqualsFinishesWithLineBreak</td>
</tr>
<tr>
<td>items</td>
<td>before words♠</td>
<td>♠ ItemStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\item♥</td>
<td>♥ ItemFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>before words♠</td>
<td>♠ SpecialBeginStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\♥</td>
<td>♥ SpecialBodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>body of special/middle★</td>
<td>★ SpecialMiddleStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\middle□</td>
<td>□ SpecialMiddleFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>body of special/middle ◆</td>
<td>◆ SpecialEndStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\♠</td>
<td>♣ SpecialEndFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>after words</td>
<td></td>
</tr>
<tr>
<td>verbatim</td>
<td>before words♠</td>
<td>♠ VerbatimBeginStartsOnOwnLine</td>
</tr>
</tbody>
</table>

---

⁸LSqB stands for Left Square Bracket
⁹LCuB stands for Left Curly Brace
6.6 Poly-switches

6.6.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches

Some poly-switches need to be partnered together; in particular, when line breaks involving the first argument of a code block need to be accounted for using both BodyStartsOnOwnLine (or its equivalent, see Table 3 on the preceding page) and LCuBStartsOnOwnLine for mandatory arguments, and LSqBStartsOnOwnLine for optional arguments.

Let’s begin with the code in Listing 457 and the YAML settings in Listing 459; with reference to Table 3 on the previous page, the key CommandNameFinishesWithLineBreak is an alias for BodyStartsOnOwnLine.

```
\mycommand
{ mand arg text
 mand arg text}
{ mand arg text
 mand arg text}
```

Listing 457: mycommand1.tex

Upon running the command

```
cmh:~$ latexindent.pl -m -l=mycom-mlb1.yaml mycommand1.tex
```

we obtain Listing 458; note that the second mandatory argument beginning brace { has had its leading line break removed, but that the first brace has not.

```
\mycommand
{ mand arg text
 mand arg text}{
 mand arg text
 mand arg text}
```

Listing 458: mycommand1.tex using Listing 459

Now let’s change the YAML file so that it is as in Listing 461; upon running the analogous command to that given above, we obtain Listing 460; both beginning braces { have had their leading line breaks removed.

```
\mycommand{
 mand arg text
 mand arg text}{
 mand arg text
 mand arg text}
```

Listing 460: mycommand1.tex using Listing 461

Now let’s change the YAML file so that it is as in Listing 463; upon running the analogous command to that given above, we obtain Listing 462.
6.6 Poly-switches

6.6.5 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches; if we use the example from Listing 457 on the preceding page, and consider the YAML settings given in Listing 465. The output from running

```bash
cmh:~$ latexindent.pl -m -l=mycom-mlb4.yaml mycommand1.tex
```

is given in Listing 465.

Studying Listing 465, we see that the two poly-switches are at opposition with one another:

- on the one hand, `LCuBStartsOnOwnLine` should not start on its own line (as poly-switch is set to `-1`);
- on the other hand, `RCuBFinishesWithLineBreak` should finish with a line break.

So, which should win the conflict? As demonstrated in Listing 464, it is clear that `LCuBStartsOnOwnLine` won this conflict, and the reason is that the second argument was processed after the first – in general, the most recently-processed code block and associated poly-switch takes priority.

We can explore this further by considering the YAML settings in Listing 467; upon running the command

```bash
cmh:~$ latexindent.pl -m -l=mycom-mlb5.yaml mycommand1.tex
```

we obtain the output given in Listing 466.

As previously, the most-recently-processed code block takes priority – as before, the second (i.e, last) argument. Exploring this further, we consider the YAML settings in Listing 469, which give associated output in Listing 468.
6.6 Poly-switches

LISTING 468: mycommand1.tex using Listing 469
\mycommand
{\text{mand arg text}} \%
{\text{mand arg text}}

Note that a % has been added to the trailing first \); this is because:

- while processing the first argument, the trailing line break has been removed (RCuBFinishesWithLineBreak set to -1);
- while processing the second argument, latexindent.pl finds that it does not begin on its own line, and so because LCuBStartsOnOwnLine is set to 2, it adds a comment, followed by a line break.

6.6.6 Conflicting poly-switches: nested code blocks

Now let's consider an example when nested code blocks have conflicting poly-switches; we'll use the code in Listing 470, noting that it contains nested environments.

LISTING 470: nested-env.tex
\begin{one}
one text
\begin{two}
two text
\end{two}
\end{one}

Let's use the YAML settings given in Listing 472, which upon running the command

cmh:$ latexindent.pl -m -l=nested-env-mlb1.yaml nested-env.tex

gives the output in Listing 471.

LISTING 471: nested-env.tex using Listing 472
\begin{one}
one text
\begin{two}
two text\end{two}\end{one}

In Listing 471, let's first of all note that both environments have received the appropriate (default) indentation; secondly, note that the poly-switch EndStartsOnOwnLine appears to have won the conflict, as \end{one} has had its leading line break removed.

To understand it, let's talk about the three basic phases of latexindent.pl:

1. Phase 1: packing, in which code blocks are replaced with unique ids, working from the inside to the outside, and then sequentially – for example, in Listing 470, the two environment is found before the one environment; if the -m switch is active, then during this phase:
   - line breaks at the beginning of the body can be added (if BodyStartsOnOwnLine is 1 or 2) or removed (if BodyStartsOnOwnLine is -1);
   - line breaks at the end of the body can be added (if EndStartsOnOwnLine is 1 or 2) or removed (if EndStartsOnOwnLine is -1);
• line breaks after the end statement can be added (if EndFinishesWithLineBreak is 1 or 2).

2. Phase 2: indentation, in which white space is added to the begin, body, and end statements;

3. Phase 3: unpacking, in which unique ids are replaced by their indented code blocks; if the -m switch is active, then during this phase,
   • line breaks before begin statements can be added or removed (depending upon BeginStartsOnOwnLine);
   • line breaks after end statements can be removed but NOT added (see EndFinishesWithLineBreak).

With reference to Listing 471, this means that during Phase 1:

• the two environment is found first, and the line break ahead of the \end{two} statement is removed because EndStartsOnOwnLine is set to −1. Importantly, because, at this stage, \end{two} does finish with a line break, EndFinishesWithLineBreak causes no action.

• next, the one environment is found; the line break ahead of \end{one} is removed because EndStartsOnOwnLine is set to −1.

The indentation is done in Phase 2; in Phase 3 there is no option to add a line break after the end statements. We can justify this by remembering that during Phase 3, the one environment will be found and processed first, followed by the two environment. If the two environment were to add a line break after the \end{two} statement, then latexindent.pl would have no way of knowing how much indentation to add to the subsequent text (in this case, \end{one}).

We can explore this further using the poly-switches in Listing 474; upon running the command

\begin{verbatim}
cmh:~$ latexindent.pl -m -l=nested-env-mlb2.yaml nested-env.tex
\end{verbatim}

we obtain the output given in Listing 473.

During Phase 1:

• the two environment is found first, and the line break ahead of the \end{two} statement is not changed because EndStartsOnOwnLine is set to 1. Importantly, because, at this stage, \end{two} does finish with a line break, EndFinishesWithLineBreak causes no action.

• next, the one environment is found; the line break ahead of \end{one} is already present, and no action is needed.

The indentation is done in Phase 2, and then in Phase 3, the one environment is found and processed first, followed by the two environment. At this stage, the two environment finds EndFinishesWithLineBreak is −1, so it removes the trailing line break; remember, at this point, latexindent.pl has completely finished with the one environment.
**SECTION 7**

The -r, -rv and -rr switches

You can instruct `latexindent.pl` to perform replacements/substitutions on your file by using any of the -r, -rv or -rr switches:

- the \texttt{-r} switch will perform indentation and replacements, not respecting verbatim code blocks;
- the \texttt{-rv} switch will perform indentation and replacements, and \textit{will} respect verbatim code blocks;
- the \texttt{-rr} switch will \textit{not} perform indentation, and will perform replacements not respecting verbatim code blocks.

We will demonstrate each of the \texttt{-r}, \texttt{-rv} and \texttt{-rr} switches, but a summary is given in Table 4.

<table>
<thead>
<tr>
<th>switch</th>
<th>indentation?</th>
<th>respect verbatim?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>-rv</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>-rr</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

The default value of the replacements field is shown in Listing 475; as with all of the other fields, you are encouraged to customise and change this as you see fit. The options in this field will only be considered if the \texttt{-r}, \texttt{-rv} or \texttt{-rr} switches are active; when discussing YAML settings related to the replacement-mode switches, we will use the style given in Listing 475.

<table>
<thead>
<tr>
<th>Listing 475: replacements</th>
</tr>
</thead>
<tbody>
<tr>
<td>replacements:</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>amalgamate: 1</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>this: 'latexindent.pl'</td>
</tr>
<tr>
<td>that: 'pl.latexindent'</td>
</tr>
<tr>
<td>lookForThis: 0</td>
</tr>
<tr>
<td>when: before</td>
</tr>
</tbody>
</table>

The first entry within the replacements field is \texttt{amalgamate}, and is \textit{optional}; by default it is set to 1, so that replacements will be amalgamated from each settings file that you specify. As you'll see in the demonstrations that follow, there is no need to specify this field.

You'll notice that, by default, there is only \texttt{one} entry in the replacements field, but it can take as many entries as you would like; each one needs to begin with a \texttt{-} on its own line.

### 7.1 Introduction to replacements

Let's explore the action of the default settings, and then we'll demonstrate the feature with further examples. With reference to Listing 475, the default action will replace every instance of the text `latexindent.pl` with `pl.latexindent`.

Beginning with the code in Listing 476 and running the command

```
`cmh:~$ latexindent.pl -r replace1.tex`
```
7.2 The two types of replacements

There are two types of replacements:

1. string-based replacements, which replace the string in `this` with the string in `that`. If you specify `this` and you do not specify `that`, then the `that` field will be assumed to be empty.

2. regex-based replacements, which use the `substitution` field.

We will demonstrate both in the examples that follow.

latexindent.pl chooses which type of replacement to make based on which fields have been specified; if the `this` field is specified, then it will make string-based replacements, regardless of if substitution is present or not.

7.3 Examples of replacements

Example 1

We begin with code given in Listing 480

```latex
\begin{env}
1 2 3\arraycolsep=3pt
4 5 6\arraycolsep=5pt
\end{env}
```

Let's assume that our goal is to remove both of the `arraycolsep` statements; we can achieve this in a few different ways.

Using the YAML in Listing 482, and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=colsep.yaml
```

then we achieve the output in Listing 481.
7.3 Examples of replacements

Note that in Listing 482, we have specified two separate fields, each with their own ‘this’ field; furthermore, for both of the separate fields, we have not specified ‘that’, so the that field is assumed to be blank by latexindent.pl;

We can make the YAML in Listing 482 more concise by exploring the substitution field. Using the settings in Listing 484 and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=colsep1.yaml
```
then we achieve the output in Listing 483.

The code given in Listing 484 is an example of a regular expression, which we may abbreviate to regex in what follows. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [12] for a detailed covering of the topic. With reference to Listing 484, we do note the following:

- the general form of the substitution field is s/regex/replacement/modifiers. You can place any regular expression you like within this;
- we have ‘escaped’ the backslash by using `\`
- we have used `\d` to represent at least one digit
- the s modifier (in the sg at the end of the line) instructs latexindent.pl to treat your file as one single line;
- the g modifier (in the sg at the end of the line) instructs latexindent.pl to make the substitution globally throughout your file; you might try removing the g modifier from Listing 484 and observing the difference in output.

You might like to see https://perldoc.perl.org/perlre.html#Modifiers for details of modifiers; in general, I recommend starting with the sg modifiers for this feature.

**Example 2** We’ll keep working with the file in Listing 480 on the preceding page for this example.

Using the YAML in Listing 486, and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=multi-line.yaml
```
then we achieve the output in Listing 485.
With reference to Listing 486, we have specified a `multi-line` version of this by employing the literal YAML style `|~`. See, for example, [https://stackoverflow.com/questions/3790454/in-yaml-how-do-i-break-a-string-over-multiple-lines](https://stackoverflow.com/questions/3790454/in-yaml-how-do-i-break-a-string-over-multiple-lines) for further options, all of which can be used in your YAML file.

This is a natural point to explore the `when` field, specified in Listing 475 on page 113. This field can take two values: `before` and `after`, which respectively instruct `latexindent.pl` to perform the replacements before indentation or after it. The default value is `before`.

Using the YAML in Listing 488, and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=multi-line1.yaml
```

then we achieve the output in Listing 487.

We note that, because we have specified `when: after`, that `latexindent.pl` has not found the string specified in Listing 488 within the file in Listing 480 on page 114. As it has looked for the string within Listing 488 after the indentation has been performed. After indentation, the string as written in Listing 488 is no longer part of the file, and has therefore not been replaced.

As a final note on this example, if you use the `-rr` switch, as follows,

```
cmh:~$ latexindent.pl -rr colsep.tex -l=multi-line1.yaml
```

then the `when` field is ignored, no indentation is done, and the output is as in Listing 485.

**Example 3** An important part of the substitution routine is in capture groups.

Assuming that we start with the code in Listing 489, let's assume that our goal is to replace each occurrence of `$$...$$` with `\begin{equation*}...\end{equation*}`. This example is partly motivated by tex stackexchange question 242150.
### 7.3 Examples of replacements

#### Listing 489: displaymath.tex

```latex
before text \$a^2+b^2=4\$ and \$c^2\$

\$
d^2+e^2 = f^2$

\$
and also \$g^2$

\$ and some inline math: \$h^2$

We use the settings in Listing 491 and run the command

cmh:\$latexindent.pl \-r \$displaymath.tex \-l=displaymath1.yaml

to receive the output given in Listing 490.
```

#### Listing 490: displaymath.tex using Listing 491

```latex
before text \begin{equation*}a^2+b^2=4\end{equation*} and \begin{equation*}c^2\end{equation*}

\begin{equation*}
d^2+e^2 = f^2
\end{equation*}

and also \begin{equation*}g^2\end{equation*} and some inline math: \$h^2$
```

#### Listing 491: displaymath1.yaml

```yaml
replacements:
- substitution: |-\s/\$/\(.*?\)\$/\begin{equation*}$1\end{equation*}/sgx
```

A few notes about Listing 491:

1. we have used the x modifier, which allows us to have white space within the regex;
2. we have used a capture group, (.*)? which captures the content between the $$...$$ into the special variable, $1;
3. we have used the content of the capture group, $1, in the replacement text.

See [https://perldoc.perl.org/perlre.html#Capture-groups](https://perldoc.perl.org/perlre.html#Capture-groups) for a discussion of capture groups.

The features of the replacement switches can, of course, be combined with others from the toolkit of latexindent.pl. For example, we can combine the poly-switches of Section 6.6 on page 96, which we do in Listing 493; upon running the command

cmh:\$latexindent.pl \-r -m \$displaymath.tex \-l=displaymath1.yaml,equation.yaml

then we receive the output in Listing 492.
Example 4  This example is motivated by tex stackexchange question 490086. We begin with the code in Listing 494.

Our goal is to make the spacing uniform between the phrases. To achieve this, we employ the settings in Listing 496, and run the command

```
\texttt{cmh:~\$ latexindent.pl \textasciitilde r phrase.tex \textasciitilde l=hspace.yaml}
```

which gives the output in Listing 495.

The \h+ setting in Listing 496 say to replace at least one horizontal space with a single space.
Our goal is to change each reference so that both the text and the reference are contained within one hyperlink. We achieve this by employing Listing 499 and running the command

```
cmh:~$ latexindent.pl -r references.tex -l=reference.yaml
```

which gives the output in Listing 498.

Referencing Listing 499, the \ means or, we have used capture groups, together with an example of an optional pattern, (?:eq)?.

**Example 6** Let's explore the three replacement mode switches (see Table 4 on page 113) in the context of an example that contains a verbatim code block, Listing 500; we will use the settings in Listing 501.

```
\begin{myenv}
body of verbatim
\end{myenv}
```

Upon running the following commands,
7.3 Examples of replacements

We note that:

1. in Listing 502 indentation has been performed, and that the replacements specified in Listing 501 have been performed, even within the verbatim code block;
2. in Listing 503 indentation has been performed, but that the replacements have not been performed within the verbatim environment, because the \texttt{rv} switch is active;
3. in Listing 504 indentation has not been performed, but that replacements have been performed, not respecting the verbatim code block.

See the summary within Table 4 on page 113.

Example 7

Let’s explore the \textit{amalgamate} field from Listing 475 on page 113 in the context of the file specified in Listing 505.

**Listing 505: amalg1.tex**

```latex
one two three
```

Let’s consider the YAML files given in Listings 506 to 508.

Upon running the following commands,

```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml
```

```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml,amalg2-yaml
```

```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml,amalg2-yaml,amalg3-yaml
```

we receive the respective output in Listings 509 to 511.
We note that:

1. in Listing 509 the replacements from Listing 506 have been used;
2. in Listing 510 the replacements from Listings 506 and 507 have both been used, because the default value of amalgamate is 1;
3. in Listing 511 only the replacements from Listing 508 have been used, because the value of amalgamate has been set to 0.
SECTION 8

The –lines switch

latexindent.pl can operate on a selection of lines of the file using the –lines or –n switch.
The basic syntax is –lines MIN–MAX, so for example

```
cmh:~$ latexindent.pl --lines 3-7 myfile.tex
```

will only operate upon lines 3 to 7 in myfile.tex. All of the other lines will not be operated upon
by latexindent.pl.

The options for the lines switch are:

- line range, as in –lines 3–7
- single line, as in –lines 5
- multiple line ranges separated by commas, as in –lines 3–5,8–10
- negated line ranges, as in –lines !3–5 which translates to –lines 1–2,6–N, where N is the
  number of lines in your file.

We demonstrate this feature, and the available variations in what follows. We will use the file in
Listing 512.

```
1 \begin{one}
2 first block, first line
3 first block, second line
4 first block, third line
5 \end{one}
6 \begin{two}
7 second block, first line
8 second block, second line
9 second block, third line
10 second block, fourth line
11 \end{two}
12 \end{one}
```

Example 8 We demonstrate the basic usage using the command

```
cmh:~$ latexindent.pl --lines 3-7 myfile.tex -o=+-mod1
```

which instructs latexindent.pl to only operate on lines 3 to 7; the output is given in Listing 513.
The following two calls to `latexindent.pl` are equivalent:

```latex
\texttt{cmh:\sim \$ latexindent.pl --lines 3-7 \texttt{myfile.tex} -o=+-mod1}
\texttt{cmh:\sim \$ latexindent.pl --lines 7-3 \texttt{myfile.tex} -o=+-mod1}
```

as `latexindent.pl` performs a check to put the lowest number first.

**Example 9**

You can call the `lines` switch with only one number and in which case only that line will be operated upon. For example

```latex
\texttt{cmh:\sim \$ latexindent.pl --lines 5 \texttt{myfile.tex} -o=+-mod2}
```

instructs `latexindent.pl` to only operate on line 5; the output is given in Listing 514.

**Listing 514: myfile-mod2.tex**

```latex
Before the environments
\begin{one}
first block, first line
first block, second line
first block, third line
\begin{two}
second block, first line
second block, second line
second block, third line
second block, fourth line
\end{two}
\end{one}
```

The following two calls are equivalent:

```latex
\texttt{cmh:\sim \$ latexindent.pl --lines 5 \texttt{myfile.tex}}
\texttt{cmh:\sim \$ latexindent.pl --lines 5-5 \texttt{myfile.tex}}
```

**Example 10**

If you specify a value outside of the line range of the file then `latexindent.pl` will ignore the `lines` argument, detail as such in the log file, and proceed to operate on the entire file.

For example, in the following call

```latex
\texttt{cmh:\sim \$ latexindent.pl --lines 11-13 \texttt{myfile.tex}}
```
latexindent.pl will ignore the lines argument, and operate on the entire file because Listing 512 only has 12 lines.

Similarly, in the call

```bash
  cmh:~$ latexindent.pl --lines -1-3 myfile.tex
```

latexindent.pl will ignore the lines argument, and operate on the entire file because we assume that negatively numbered lines in a file do not exist.

**Example 11** You can specify *multiple line ranges* as in the following

```bash
  cmh:~$ latexindent.pl --lines 3-5,8-10 myfile.tex -o=+-mod3
```

which instructs latexindent.pl to operate upon lines 3 to 5 and lines 8 to 10; the output is given in Listing 515.

**LISTING 515: myfile-mod3.tex**

```latex
  1 Before the environments
  2 \begin{one}
  3 first block, first line
  4 first block, second line
  5 first block, third line
  6 \begin{two}
  7 second block, first line
  8 second block, second line
  9 second block, third line
 10 second block, fourth line
11 \end{two}
12 \end{one}
```

The following calls to latexindent.pl are all equivalent

```bash
  cmh:~$ latexindent.pl --lines 3-5,8-10 myfile.tex
  cmh:~$ latexindent.pl --lines 8-10,3-5 myfile.tex
  cmh:~$ latexindent.pl --lines 10-8,3-5 myfile.tex
  cmh:~$ latexindent.pl --lines 10-8,5-3 myfile.tex
```

as latexindent.pl performs a check to put the lowest line ranges first, and within each line range, it puts the lowest number first.

**Example 12** There’s no limit to the number of line ranges that you can specify, they just need to be separated by commas. For example

```bash
  cmh:~$ latexindent.pl --lines 1-2,4-5,9-10,12 myfile.tex -o=+-mod4
```

has four line ranges: lines 1 to 2, lines 4 to 5, lines 9 to 10 and line 12. The output is given in Listing 516.
As previously, the ordering does not matter, and the following calls to \texttt{latexindent.pl} are all equivalent:

\begin{verbatim}
$ latexindent.pl --lines 1-2,4-5,9-10,12 myfile.tex
$ latexindent.pl --lines 2-1,4-5,9-10,12 myfile.tex
$ latexindent.pl --lines 4-5,1-2,9-10,12 myfile.tex
$ latexindent.pl --lines 12,4-5,1-2,9-10 myfile.tex
\end{verbatim}

You can specify \texttt{negated line ranges} by using \texttt{!} as in

\begin{verbatim}
$ latexindent.pl --lines !5-7 myfile.tex -o=+-mod5
\end{verbatim}

which instructs \texttt{latexindent.pl} to operate upon all of the lines except lines 5 to 7.

In other words, \texttt{latexindent.pl} \textit{will} operate on lines 1 to 4, and 8 to 12, so the following two calls are equivalent:

\begin{verbatim}
$ latexindent.pl --lines !5-7 myfile.tex
$ latexindent.pl --lines 1-4,8-12 myfile.tex
\end{verbatim}

The output is given in Listing 517.
Example 14  You can specify multiple negated line ranges such as

```bash
$ latexindent.pl --lines !5-7,!9-10 myfile.tex -o=+-mod6
```

which is equivalent to:

```bash
$ latexindent.pl --lines 1-4,8,11-12 myfile.tex -o=+-mod6
```

The output is given in Listing 518.

```
Listing 518: myfile-mod6.tex

\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}

\begin{two}
  second block, first line
  second block, second line
  second block, third line
  second block, fourth line
\end{two}
```

Example 15  If you specify a line range with anything other than an integer, then `latexindent.pl` will ignore the `lines` argument, and operate on the entire file.

Sample calls that result in the `lines` argument being ignored include the following:

```bash
$ latexindent.pl --lines 1-x myfile.tex
$ latexindent.pl --lines !y-3 myfile.tex
```

Example 16  We can, of course, use the `lines` switch in combination with other switches.

For example, let's use with the file in Listing 519.

```
Listing 519: myfile1.tex

\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}

\begin{two}
  body \end{two}
\end{one}
```

We can demonstrate interaction with the `-m` switch (see Section 6 on page 68); in particular, if we use Listing 414 on page 100, Listing 398 on page 99 and Listing 399 on page 99 and run

```bash
$ latexindent.pl --lines 6 myfile1.tex -o=+-mod1 -m -l env-mlb2,env-mlb7,env-mlb8 -o=+-mod1
```

then we receive the output in Listing 520.
Before the environments
\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}
\begin{two}
  body
\end{two}
\end{one}
SECTION 9

Fine tuning

latexindent.pl operates by looking for the code blocks detailed in Table 2 on page 45. The fine tuning of the details of such code blocks is controlled by the `fineTuning` field, detailed in Listing 521. This field is for those that would like to peek under the bonnet/hood and make some fine tuning to latexindent.pl's operating.

**Warning!**
Making changes to the fine tuning may have significant consequences for your indentation scheme, proceed with caution!

Listing 521: fineTuning

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>630</td>
<td><code>fineTuning:</code></td>
</tr>
<tr>
<td>631</td>
<td><code>environments:</code></td>
</tr>
<tr>
<td>632</td>
<td><code>name: '[a-zA-Z0]*0-9_\]'+</code></td>
</tr>
<tr>
<td>633</td>
<td><code>ifElseFi:</code></td>
</tr>
<tr>
<td>634</td>
<td><code>name: '([^\?\?if[a-zA-Z0]*\?\{\}0?if[a-zA-Z0]*\?\}'</code></td>
</tr>
<tr>
<td>635</td>
<td><code>commands:</code></td>
</tr>
<tr>
<td>636</td>
<td><code>name: '[+a-zA-Z0]*0-9_\]+</code></td>
</tr>
<tr>
<td>637</td>
<td><code>items:</code></td>
</tr>
<tr>
<td>638</td>
<td>`canBeFollowedBy: '(?&lt;![-]*{]?</td>
</tr>
<tr>
<td>639</td>
<td><code>keyEqualsValuesBracesBrackets:</code></td>
</tr>
<tr>
<td>640</td>
<td><code>name: '[a-zA-Z0]*0-9_\][\-]++[a-zA-Z0]*0-9_\].h{\}:\#]*'</code></td>
</tr>
<tr>
<td>641</td>
<td>`follow: '(?&lt;!{}?</td>
</tr>
<tr>
<td>642</td>
<td><code>namedGroupingBracesBrackets:</code></td>
</tr>
<tr>
<td>643</td>
<td><code>name: '[0-9\._a-zA-Z0]*\[\]+'</code></td>
</tr>
<tr>
<td>644</td>
<td><code>follow: '\h\{\}[\]{\}[\}\}'</code></td>
</tr>
<tr>
<td>645</td>
<td><code>UnNamedGroupingBracesBrackets:</code></td>
</tr>
<tr>
<td>646</td>
<td><code>follow: '\{\}\{\}h{\}h{\}'</code></td>
</tr>
<tr>
<td>647</td>
<td><code>arguments:</code></td>
</tr>
<tr>
<td>648</td>
<td>`before: '(?:#d\h*;?,?+/?)+</td>
</tr>
<tr>
<td>649</td>
<td><code>between: '_\{-\}h*\}'</code></td>
</tr>
<tr>
<td>650</td>
<td><code>trailingComments:</code></td>
</tr>
<tr>
<td>651</td>
<td><code>notPreceededBy: '(?&lt;&lt;!\)'</code></td>
</tr>
<tr>
<td>652</td>
<td><code>modifyLineBreaks:</code></td>
</tr>
<tr>
<td>653</td>
<td><code>betterFullStop:</code></td>
</tr>
<tr>
<td>654</td>
<td><code>doubleBackSlash: '\\\\(?!h[a-zA-Z]\)\{?.&lt;!(?&lt;:(?:e\.)\{??:i.e\}\{??:etc\})))\{?!h[a-zA-Z]\}h[\-\{\}h[0-9])\}'</code></td>
</tr>
<tr>
<td>655</td>
<td><code>comma: ','</code></td>
</tr>
</tbody>
</table>

The fields given in Listing 521 are all regular expressions. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [12] for a detailed covering of the topic.

We make the following comments with reference to Listing 521:

1. the `environments:name` field details that the `name` of an environment can contain:
   (a) a-z lower case letters
   (b) A-Z upper case letters
   (c) @ the @ 'letter'
The + at the end means at least one of the above characters.

2. the `ifElseFi:name` field:
   (a) @? means that it can possibly begin with @
   (b) followed by if
   (c) followed by 0 or more characters from a-z, A-Z and @
   (d) the ? at the end means non-greedy, which means 'stop the match as soon as possible'

3. the `keyEqualsValuesBracesBrackets` contains some interesting syntax:
   (a) | means 'or'
   (b) (?:(?!\){) the (?:...) uses a non-capturing group – you don’t necessarily need to worry about what this means, but just know that for the fineTuning feature you should only ever use non-capturing groups, and not capturing groups, which are simply (...)
   (c) (?<!\){) means a { but it can not be immediately preceded by a \n
4. in the `arguments:before` field
   (a) \d\h* means a digit (i.e. a number), followed by 0 or more horizontal spaces
   (b) ;?,? means possibly a semi-colon, and possibly a comma
   (c) \<.*?\> is designed for 'beamer'-type commands; the .*? means anything in between <...>

5. the `modifyLineBreaks` field refers to fine tuning settings detailed in Section 6 on page 68. In particular:
   (a) `betterFullStop` is in relation to the one sentence per line routine, detailed in Section 6.5 on page 88
   (b) `doubleBackSlash` is in relation to the `DBSStartsOnOwnLine` and `DBSFinishesWithLineBreak` polyswitches surrounding double back slashes, see Section 6.6.2 on page 104
   (c) `comma` is in relation to the `CommaStartsOnOwnLine` and `CommaFinishesWithLineBreak` polyswitches surrounding commas in optional and mandatory arguments; see Table 3 on page 108

It is not obvious from Listing 521, but each of the follow, before and between fields allow trailing comments, line breaks, and horizontal spaces between each character.

**Warning!**
For the fineTuning feature you should only ever use non-capturing groups, such as (?:...) and not capturing groups, which are (...)

**Example 17**
As a demonstration, consider the file given in Listing 522, together with its default output using the command

```
$ latexindent.pl finetuning1.tex
```

is given in Listing 523.
It's clear from Listing 523 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 525 and running the command

```bash
cmh:~$ latexindent.pl finetuning1.tex -l=fine-tuning1.yaml
```

and the associated (desired) output is given in Listing 524.

**Example 18**

Let's have another demonstration; consider the file given in Listing 526, together with its default output using the command

```bash
cmh:~$ latexindent.pl finetuning2.tex
```

is given in Listing 527.

It's clear from Listing 527 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 529 and running the command

```bash
cmh:~$ latexindent.pl finetuning2.tex -l=fine-tuning2.yaml
```

and the associated (desired) output is given in Listing 528.
In particular, note that the settings in Listing 529 specify that NamedGroupingBracesBrackets and UnNamedGroupingBracesBrackets can follow " and that we allow --- between arguments.

Example 19 You can tweak the fineTuning using the -y switch, but to be sure to use quotes appropriately. For example, starting with the code in Listing 530 and running the following command

```
cmh::~$ latexindent.pl -m
    -y='modifyLineBreaks:oneSentencePerLine:manipulateSentences:a-z:1,  
    modifyLineBreaks:oneSentencePerLine:sentencesBeginWith:a-z:1,  
    fineTuning:modifyLineBreaks:betterFullStop:  
    "(?:\.|;|:(?!(?:(?:e.g)|(?:i.e)|(?:etc))))\.|(?!(?:[a-z]|[A-Z]|\-|~|,|[0-9]))" 
issue-243.tex -o=+-mod1
```

gives the output shown in Listing 531.

Example 20 We can tweak the fineTuning for how trailing comments are classified. For motivation, let's consider the code given in Listing 532

```
\href{Handbook%20for%30Spoken%40document.pdf}{my document}
```

We will compare the settings given in Listings 533 and 534.
Upon running the following commands

```latex
\$ latexindent.pl -m finetuning4.tex -o=+-mod1 -l=href1
\$ latexindent.pl -m finetuning4.tex -o=+-mod2 -l=href2
```

we receive the respective output in Listings 535 and 536.

<table>
<thead>
<tr>
<th>Listing 535: finetuning4.tex using Listing 533</th>
</tr>
</thead>
<tbody>
<tr>
<td>some before text \href{Handbook some after text%20for%30Spoken%40document.pdf}{my document}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 536: finetuning4.tex using Listing 534</th>
</tr>
</thead>
<tbody>
<tr>
<td>some before text \href{Handbook%20for%30Spoken%40document.pdf}{my document} some after text</td>
</tr>
</tbody>
</table>

We note that in:

- Listing 535 the trailing comments are assumed to be everything following the first comment symbol, which has meant that everything following it has been moved to the end of the line; this is undesirable, clearly!

- Listing 536 has fine-tuned the trailing comment matching, and says that % cannot be immediately preceeded by the words ‘Handbook’, ‘for’ or ‘Spoken’, which means that none of the % symbols have been treated as trailing comments, and the output is desirable.

Another approach to this situation, which does not use fineTuning, is to use noIndentBlock which we discussed in Listing 24 on page 23; using the settings in Listing 537 and running the command

```latex
\$ latexindent.pl -m finetuning4.tex -o=+-mod3 -l=href3
```

then we receive the same output given in Listing 536; see also paragraphsStopAt in Listing 319 on page 83.

<table>
<thead>
<tr>
<th>Listing 537: href3.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 80</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>removeParagraphLineBreaks:</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
<tr>
<td>paragraphsStopAt:</td>
</tr>
<tr>
<td>verbatim: 0</td>
</tr>
<tr>
<td>noIndentBlock:</td>
</tr>
<tr>
<td>href:</td>
</tr>
<tr>
<td>begin: '\href{[^}]*?}{'</td>
</tr>
<tr>
<td>body: '[^}]*?'</td>
</tr>
<tr>
<td>end: '}}'</td>
</tr>
</tbody>
</table>

With reference to the body field in Listing 537, we note that the body field can be interpreted as: the fewest number of zero or more characters that are not right braces. This is an example of character class.

**Example 21** We can use the fineTuning field to assist in the formatting of bibliography files.

Starting with the file in Listing 538 and running the command

```latex
\$ latexindent.pl -m finetuning4.tex -o=+-mod4 -l=href4
```
gives the output in Listing 539.

<table>
<thead>
<tr>
<th>LISTING 538: bib1.bib</th>
<th>LISTING 539: bib1-mod1.bib</th>
</tr>
</thead>
<tbody>
<tr>
<td>@online{paulo,</td>
<td>@online{paulo,</td>
</tr>
<tr>
<td>title = &quot;arararule,</td>
<td>title = &quot;arararule,</td>
</tr>
<tr>
<td>indent.yaml&quot;,</td>
<td>indent.yaml&quot;,</td>
</tr>
<tr>
<td>author = &quot;PauloCereda&quot;,</td>
<td>author = &quot;PauloCereda&quot;,</td>
</tr>
<tr>
<td>date = {2013-05-23},</td>
<td>date = {2013-05-23},</td>
</tr>
<tr>
<td>urldate = {2021-03-19},</td>
<td>urldate = {2021-03-19},</td>
</tr>
<tr>
<td>keywords = {contributor},</td>
<td>keywords = {contributor},</td>
</tr>
</tbody>
</table>

Let's assume that we would like to format the output so as to align the = symbols. Using the settings in Listing 541 and running the command

```
cmh:~$ latexindent.pl bib1.bib -l bibsettings1.yaml -o=+-mod2
```

gives the output in Listing 540.

<table>
<thead>
<tr>
<th>LISTING 540: bib1.bib using Listing 541</th>
</tr>
</thead>
<tbody>
<tr>
<td>@online{paulo,</td>
</tr>
<tr>
<td>title = &quot;arararule, indent.yaml&quot;,</td>
</tr>
<tr>
<td>author = &quot;PauloCereda&quot;,</td>
</tr>
<tr>
<td>date = {2013-05-23},</td>
</tr>
<tr>
<td>urldate = {2021-03-19},</td>
</tr>
<tr>
<td>keywords = {contributor},</td>
</tr>
</tbody>
</table>

Some notes about Listing 541:

- we have populated the `lookForAlignDelims` field with the `online` command, and have used the `delimiterRegEx` discussed in Section 5.5.4 on page 35;
- we have tweaked the `keyEqualsValuesBracesBrackets` code block so that it will not be found following a comma; this means that, in contrast to the default behaviour, the lines such as date={2013-05-23}, will not be treated as key-equals-value braces;
- the adjustment to `keyEqualsValuesBracesBrackets` necessitates the associated change to the `UnNamedGroupingBracesBrackets` field so that they will be searched for following = symbols.

**Example 22** We can build upon Listing 541 for slightly more complicated bibliography files.

Starting with the file in Listing 542 and running the command

```
cmh:~$ latexindent.pl bib2.bib -l bibsettings1.yaml -o=+-mod1
```

gives the output in Listing 543.
The output in Listing 543 is not ideal, as the = symbol within the url field has been incorrectly used as an alignment delimiter.

We address this by tweaking the delimiterRegEx field in Listing 544.

Upon running the command

```
cmh:~\$ latexindent.pl bib2.bib -l bibsettings1.yaml,bibsettings2.yaml -o=-mod2
```

we receive the desired output in Listing 545.

With reference to Listing 544 we note that the delimiterRegEx has been adjusted so that = symbols are used as the delimiter, but only when they are not preceded by either v or spfreload.
Conclusions and known limitations

There are a number of known limitations of the script, and almost certainly quite a few that are unknown!

For example, with reference to the multicolumn alignment routine in Listing 52 on page 29, when working with code blocks in which multicolumn commands overlap, the algorithm can fail.

Another limitation is to do with efficiency, particularly when the -m switch is active, as this adds many checks and processes. The current implementation relies upon finding and storing every code block (see the discussion on page 111); I hope that, in a future version, only nested code blocks will need to be stored in the 'packing' phase, and that this will improve the efficiency of the script.

You can run \texttt{latexindent} on any file; if you don't specify an extension, then the extensions that you specify in \texttt{fileExtensionPreference} (see Listing 16 on page 20) will be consulted. If you find a case in which the script struggles, please feel free to report it at [13], and in the meantime, consider using a \texttt{noIndentBlock} (see page 23).

I hope that this script is useful to some; if you find an example where the script does not behave as you think it should, the best way to contact me is to report an issue on [13]; otherwise, feel free to find me on the \url{http://tex.stackexchange.com/users/6621/cmhughes}.
11.1 External links

6. Data::Dumper module. URL: https://perldoc.perl.org/Data::Dumper (visited on 06/18/2021).
12. perldoc Encode::Supported. URL: https://perldoc.perl.org/Encode::Supported (visited on 05/06/2021).
16. Video demonstration of latexindent.pl on youtube. URL: https://www.youtube.com/watch?v=Wo0E93GA050 (visited on 01/01/2022).

11.2 Contributors

1. Sam Abey. Print usage tip to STDERR only if STDIN is TTY. Sept. 17, 2019. URL: https://github.com/cmhughes/latexindent.pl/pull/174 (visited on 03/19/2021).


SECTION A

Required Perl modules

If you intend to use latexindent.pl and not one of the supplied standalone executable files, then you will need a few standard Perl modules – if you can run the minimum code in Listing 546 (perl helloworld.pl) then you will be able to run latexindent.pl, otherwise you may need to install the missing modules – see appendices A.1 and A.2.

### Listing 546: helloworld.pl

```perl
#!/usr/bin/perl
use strict;
use warnings;
use utf8;
use PerlIO::encoding;
use Unicode::GCString;
use open ':std', ':encoding(UTF-8)';
use Text::Wrap;
use Text::Tabs;
use FindBin;
use YAML::Tiny;
use File::Copy;
use File::Basename;
use File::HomeDir;
use Encode;
use Getopt::Long;
use Data::Dumper;
use List::Util qw(max);

print "hello world";
exit;
```

A.1 Module installer script

latexindent.pl ships with a helper script that will install any missing perl modules on your system; if you run

```
cmh:~$ perl latexindent-module-installer.pl
```

or

```
C:\Users\cmh>perl latexindent-module-installer.pl
```

then, once you have answered Y, the appropriate modules will be installed onto your distribution.

A.2 Manually installing modules

Manually installing the modules given in Listing 546 will vary depending on your operating system and Perl distribution.
A.2 Manually installing modules

A.2.1 Linux

A.2.1.1 perlbrew

Linux users may be interested in exploring Perlbrew [23]; an example installation would be:

```bash
$ sudo apt-get install perlbrew
$ perlbrew init
$ perlbrew install perl-5.28.1
$ perlbrew switch perl-5.28.1
$ sudo apt-get install curl
$ curl -L http://cpanmin.us | perl - App::cpanminus
$ cpanm YAML::Tiny
$ cpanm File::HomeDir
$ cpanm Unicode::GCString
```

A.2.1.2 Ubuntu/Debian

For other distributions, the Ubuntu/Debian approach may work as follows:

```bash
$ sudo apt install perl
$ sudo cpan -i App::cpanminus
$ sudo cpanm YAML::Tiny
$ sudo cpanm File::HomeDir
$ sudo cpanm Unicode::GCString
```

or else by running, for example,

```bash
$ sudo perl -MCPAN -e'install "File::HomeDir"
```

A.2.1.3 Ubuntu: using the texlive from apt-get

Ubuntu users that install texlive using apt-get as in the following

```bash
$ sudo apt install texlive
$ sudo apt install texlive-latex-recommended
```

may need the following additional command to work with `latexindent.pl`

```bash
$ sudo apt install texlive-extra-utils
```

A.2.1.4 Alpine

If you are using Alpine, some Perl modules are not build-compatible with Alpine, but replacements are available through `apk`. For example, you might use the commands given in Listing 547; thanks to [15] for providing these details.
A.2 Manually installing modules

Listing 547: alpine-install.sh

# Installing perl
apk --no-cache add miniperl perl-utils

# Installing incompatible latexindent perl dependencies via apk
apk --no-cache add perl-log-dispatch
perl-namespace-autoclean
perl-specio
perl-unicode-linebreak

# Installing remaining latexindent perl dependencies via cpan
apk --no-cache add curl wget make
ls /usr/share/texmf-dist/scripts/latexindent
cd /usr/local/bin && 
curl -L https://cpanmin.us/ -o cpanm && 
chmod +x cpanm
cpanm -n App::cpanminus
cpanm -n File::HomeDir
cpanm -n Params::ValidationCompiler
cpanm -n YAML::Tiny
cpanm -n Unicode::GCString

Users of NixOS might like to see https://github.com/cmhughes/latexindent.pl/issues/222 for tips.

A.2.2 Mac

Users of the Macintosh operating system might like to explore the following commands, for example:

```shell
$ brew install perl
$ brew install cpanm
$ cpanm YAML::Tiny
$ cpanm File::HomeDir
$ cpanm Unicode::GCString
```

A.2.3 Windows

Strawberry Perl users on Windows might use CPAN client. All of the modules are readily available on CPAN [7].

indent.log will contain details of the location of the Perl modules on your system. latexindent.exe is a standalone executable for Windows (and therefore does not require a Perl distribution) and caches copies of the Perl modules onto your system; if you wish to see where they are cached, use the trace option, e.g

```shell
C:\Users\cmh>latexindent.exe -t myfile.tex
```
SECTION B

Updating the path variable

\texttt{latexindent.pl} has a few scripts (available at [13]) that can update the path variables. Thank you to [17] for this feature. If you're on a Linux or Mac machine, then you'll want \texttt{CMakeLists.txt} from [13].

B.1 Add to path for Linux

To add \texttt{latexindent.pl} to the path for Linux, follow these steps:

1. download \texttt{latexindent.pl} and its associated modules, \texttt{defaultSettings.yaml}, to your chosen directory from [13];
2. within your directory, create a directory called \texttt{path-helper-files} and download \texttt{CMakeLists.txt} and \texttt{cmake_uninstall.cmake.in} from [13]/\texttt{path-helper-files} to this directory;
3. run

\begin{verbatim}
cmh:~$ ls /usr/local/bin
\end{verbatim}

to see what is \textit{currently} in there;
4. run the following commands

\begin{verbatim}
cmh:~$ sudo apt-get install cmake
cmh:~$ sudo apt-get update && sudo apt-get install build-essential
cmh:~$ mkdir build && cd build
cmh:~$ cmake ../path-helper-files
cmh:~$ sudo make install
\end{verbatim}

5. run

\begin{verbatim}
cmh:~$ ls /usr/local/bin
\end{verbatim}

again to check that \texttt{latexindent.pl}, its modules and \texttt{defaultSettings.yaml} have been added.

To remove the files, run

\begin{verbatim}
cmh:~$ sudo make uninstall
\end{verbatim}

B.2 Add to path for Windows

To add \texttt{latexindent.exe} to the path for Windows, follow these steps:

1. download \texttt{latexindent.exe}, \texttt{defaultSettings.yaml}, \texttt{add-to-path.bat} from [13] to your chosen directory;
2. open a command prompt and run the following command to see what is \textit{currently} in your \texttt{%path%} variable;
3. right click on add-to-path.bat and Run as administrator;
4. log out, and log back in;
5. open a command prompt and run

```
C:\Users\cmh> echo %path%
```

to check that the appropriate directory has been added to your %path%.

To remove the directory from your %path%, run remove-from-path.bat as administrator.
latexindent-yaml-schema.json

latexindent.pl ships with latexindent-yaml-schema.json which might help you when constructing your YAML files.

C.1 VSCode demonstration

To use latexindent-yaml-schema.json with VSCode, you can use the following steps:

1. download latexindent-yaml-schema.json from the documentation folder of [13], save it in whichever directory you would like, noting it for reference;
2. following the instructions from [14], for example, you should install the VSCode YAML extension [32];
3. set up your settings.json file using the directory you saved the file by adapting Listing 548; on my Ubuntu laptop this file lives at /home/cmhughes/.config/Code/User/settings.json.

```json
// LISTING 548: settings.json
{
  "yamlschemas": {
    "/home/cmhughes/projects/latexindent/documentation/latexindent-yaml-schema.json": 
    "/home/cmhughes/projects/latexindent/defaultSettings.yaml"
  },
  "redhat.telemetry.enabled": true
}
```

Alternatively, if you would prefer not to download the json file, you might be able to use an adapted version of Listing 549.

```json
// LISTING 549: settings-alt.json
{
  "yamlschemas": {
    "/home/cmhughes/projects/latexindent/defaultSettings.yaml"
  }
}
```

Finally, if your TeX distribution is up to date, then latexindent-yaml-schema.json should be in the documentation folder of your installation, so an adapted version of Listing 550 may work.

```json
// LISTING 550: settings-alt1.json
{
  "yamlschemas": {
    "/usr/local/texlive/2021/texmf-dist/doc/support/latexindent/latexindent-yaml-schema.json": 
    "/home/cmhughes/projects/latexindent/defaultSettings.yaml"
  }
}
```

If you have details of how to implement this schema in other editors, please feel encouraged to contribute to this documentation.
SECTION D

Using conda

If you use conda you'll only need

```bash
cmh:~$ conda install latexindent.pl -c conda-forge
```

this will install the executable and all its dependencies (including perl) in the activate environment.

You don't even have to worry about defaultSettings.yaml as it included too, you can thus skip appendices A and B.

You can get a conda installation for example from [6] or from [3].
SECTION E

logFilePreferences

Listing 17 on page 21 describes the options for customising the information given to the log file, and we provide a few demonstrations here. Let's say that we start with the code given in Listing 551, and the settings specified in Listing 552.

<table>
<thead>
<tr>
<th>LISTING 551: simple.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>body of myenv</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 552: logfile-prefs1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>logFilePreferences:</td>
</tr>
<tr>
<td>showDecorationStartCodeBlockTrace: &quot;+++++&quot;);</td>
</tr>
<tr>
<td>showDecorationFinishCodeBlockTrace: &quot;-----&quot;</td>
</tr>
</tbody>
</table>

If we run the following command (noting that -t is active)

```plaintext
cmh:~$ latexindent.pl -t -l=logfile-prefs1.yaml simple.tex
```

then on inspection of indent.log we will find the snippet given in Listing 553.

<table>
<thead>
<tr>
<th>LISTING 553: indent.log</th>
</tr>
</thead>
<tbody>
<tr>
<td>++++</td>
</tr>
<tr>
<td>TRACE: environment found: myenv</td>
</tr>
<tr>
<td>No ancestors found for myenv</td>
</tr>
<tr>
<td>Storing settings for myenv environments</td>
</tr>
<tr>
<td>indentRulesGlobal specified (0) for environments, ...</td>
</tr>
<tr>
<td>Using defaultIndent for myenv</td>
</tr>
<tr>
<td>Putting linebreak after replacementText for myenv</td>
</tr>
<tr>
<td>looking for COMMANDS and key = {value}</td>
</tr>
<tr>
<td>TRACE: Searching for commands with optional and/or mandatory arguments AND key = {value}</td>
</tr>
<tr>
<td>looking for SPECIAL begin/end</td>
</tr>
<tr>
<td>TRACE: Searching myenv for special begin/end (see specialBeginEnd)</td>
</tr>
<tr>
<td>TRACE: Searching myenv for optional and mandatory arguments</td>
</tr>
<tr>
<td>... no arguments found</td>
</tr>
<tr>
<td>-----</td>
</tr>
</tbody>
</table>

Notice that the information given about myenv is ‘framed’ using ++++ and ----- respectively.
In relation to Section 4 on page 16, Windows users that encounter encoding issues with `indentconfig.yaml`, may wish to run the following command in either `cmd.exe` or `powershell.exe`:

```bash
C:\Users\cmh>chcp
```

They may receive the following result:

```bash
C:\Users\cmh>Active code page: 936
```

and can then use the settings given in Listing 554 within their `indentconfig.yaml`, where 936 is the result of the `chcp` command.

```
LISTING 554: encoding demonstration for indentconfig.yaml
encoding: cp936
```
SECTION G

dos2unix linebreak adjustment

dos2unixlinebreaks: \{integer\}

If you use latexindent.pl on a dos-based Windows file on Linux then you may find that trailing horizontal space is not removed as you hope.

In such a case, you may wish to try setting dos2unixlinebreaks to 1 and employing, for example, the following command.

```
cmh:~$ latexindent.pl -y="dos2unixlinebreaks:1" myfile.tex
```

See [33] for further details.
SECTION H

Differences from Version 2.2 to 3.0

There are a few (small) changes to the interface when comparing Version 2.2 to Version 3.0. Explicitly, in previous versions you might have run, for example,

```bash
~$ latexindent.pl -o myfile.tex outputfile.tex
```

whereas in Version 3.0 you would run any of the following, for example,

```bash
~$ latexindent.pl -o=outputfile.tex myfile.tex
~$ latexindent.pl -o outputfile.tex myfile.tex
~$ latexindent.pl myfile.tex -o outputfile.tex
~$ latexindent.pl myfile.tex -o=outputfile.tex
~$ latexindent.pl myfile.tex -outputfile=outputfile.tex
```

noting that the output file is given next to the -o switch.

The fields given in Listing 555 are obsolete from Version 3.0 onwards.

<table>
<thead>
<tr>
<th>Listing 555: Obsolete YAML fields from Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>alwaysLookforSplitBrackets</td>
</tr>
<tr>
<td>alwaysLookforSplitBrackets</td>
</tr>
<tr>
<td>checkunmatched</td>
</tr>
<tr>
<td>checkunmatchedELSE</td>
</tr>
<tr>
<td>checkunmatchedbracket</td>
</tr>
<tr>
<td>constructIfElseFi</td>
</tr>
</tbody>
</table>

There is a slight difference when specifying indentation after headings; specifically, we now write indentAfterThisHeading instead of indent. See Listings 556 and 557.

<table>
<thead>
<tr>
<th>Listing 556: indentAfterThisHeading in Version 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
</tr>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indent: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 557: indentAfterThisHeading in Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
</tr>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indentAfterThisHeading: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>

To specify noAdditionalIndent for display-math environments in Version 2.2, you would write YAML as in Listing 558; as of Version 3.0, you would write YAML as in Listing 559 or, if you're using -m switch, Listing 560.
**LISTING 558: noAdditionalIndent in Version 2.2**

noAdditionalIndent:
\[: 0
\]: 0

---

**LISTING 559: noAdditionalIndent for displayMath in Version 3.0**

specialBeginEnd:
  displayMath:
    begin: '\\['
    end: '\\']'
    lookForThis: 0

**LISTING 560: noAdditionalIndent for displayMath in Version 3.0**

noAdditionalIndent:
  displayMath: 1
Listings

Listing 1: demo-tex.tex ........................................ 5
Listing 2: fileExtensionPreference .................. 5
Listing 3: modifyLineBreaks ......................... 5
Listing 4: replacements ................................ 5
Listing 5: Possible error messages ................. 6
Listing 6: filecontents1.tex ......................... 7
Listing 7: filecontents1.tex default output .... 7
Listing 8: tikzset.tex ................................ 7
Listing 9: tikzset.tex default output .......... 7
Listing 10: pstricks.tex ................................ 7
Listing 11: pstricks.tex default output ....... 7
Listing 14: The encoding option for indentconfig.yaml ........................................ 17
Listing 16: fileExtensionPreference ............ 20
Listing 17: logFilePreferences ................... 21
Listing 18: verbatimEnvironments ................. 22
Listing 19: verbatimCommands ...................... 22
Listing 20: nameAsRegex1.yaml .................... 22
Listing 21: nameAsRegex2.yaml .................... 22
Listing 22: nameAsRegex3.yaml .................... 23
Listing 23: nameAsRegex4.yaml .................... 23
Listing 24: noIndentBlock ......................... 23
Listing 25: noIndentBlock.tex ...................... 23
Listing 26: noIndentBlock1.tex .................... 23
Listing 27: noindent1.yaml .......................... 24
Listing 28: noindent2.yaml ......................... 24
Listing 29: noindent3.yaml ......................... 24
Listing 30: noIndentBlock1.tex using Listing 27
or Listing 28 .............................................. 24
Listing 31: noIndentBlock1.tex using Listing 29
or Listing 28 .............................................. 24
Listing 32: nameAsRegex5.yaml .................... 25
Listing 33: nameAsRegex6.yaml .................... 25
Listing 34: fileContentsEnvironments ............ 25
Listing 35: lookForPreamble ......................... 25
Listing 36: Motivating preambleCommandsBeforeEnvironments .................................... 26
Listing 37: removeTrailingWhitespace ............. 26
Listing 40: tabular1.tex ............................ 27
Listing 41: tabular1.tex default output .......... 27
Listing 42: lookForAlignDelims (advanced) ...... 27
Listing 43: tabular2.tex ............................ 28
Listing 44: tabular2.yaml ............................ 28
Listing 45: tabular3.yaml ............................ 28
Listing 46: tabular4.yaml ............................ 28
Listing 47: tabular5.yaml ............................ 28
Listing 48: tabular6.yaml ............................ 28
Listing 49: tabular7.yaml ............................ 28
Listing 50: tabular8.yaml ............................ 28
Listing 51: tabular2.tex default output ....... 29
Listing 52: tabular2.tex using Listing 44 ...... 29
Listing 53: tabular2.tex using Listing 45 ...... 29
Listing 54: tabular2.tex using Listings 44 and 46 29
Listing 55: tabular2.tex using Listings 44 and 47 30
Listing 56: tabular2.tex using Listings 44 and 48 30
Listing 57: tabular2.tex using Listings 44 and 49 30
Listing 58: tabular2.tex using Listings 44 and 50 30
Listing 59: aligned1.tex ............................ 31
Listing 60: aligned1-default.tex .................. 31
Listing 61: sba1.yaml .................................. 31
Listing 62: sba2.yaml .................................. 31
Listing 63: sba3.yaml .................................. 31
Listing 64: sba4.yaml .................................. 31
Listing 65: aligned1-mod1.tex ...................... 32
Listing 66: sba5.yaml .................................. 32
Listing 67: sba6.yaml .................................. 32
Listing 68: aligned1-mod5.tex ...................... 32
Listing 69: aligned1.tex using Listing 70 ...... 32
Listing 70: sba7.yaml .................................. 32
Listing 71: tabular4.tex ............................ 33
Listing 72: tabular4-default.tex .................. 33
Listing 73: tabular4-FDBS.tex ...................... 33
Listing 74: matrix1.tex ................................ 33
Listing 75: matrix1.tex default output .......... 33
Listing 76: align-block.tex ......................... 33
Listing 77: align-block.tex default output ...... 33
Listing 78: tabular-DM.tex .......................... 34
Listing 79: tabular-DM.tex default output ...... 34
Listing 80: tabular-DM.tex using Listing 81 ... 34
Listing 81: dontMeasure1.yaml ..................... 34
Listing 82: tabular-DM.tex using Listing 83 or Listing 85 ........................................ 34
Listing 83: dontMeasure2.yaml ..................... 34

[git]  •  main @ 45f6411  •  2022-01-02  •  V3.13.5
LISTING 551: `simple.tex` ................................. 145
LISTING 552: `logfile-prefs1.yaml` ....................... 145
LISTING 553: `indent.log` ................................. 145
LISTING 554: encoding demonstration for `indentconfig.yaml` .................................................. 146
LISTING 555: Obsolete YAML fields from Version 3.0 ...... 148
LISTING 556: `indentAfterThisHeading` in Version 2.2 .... 148

LISTING 557: `indentAfterThisHeading` in Version 3.0 ................................................................. 148
LISTING 558: `noAdditionalIndent` in Version 2.2 ...... 149
LISTING 559: `noAdditionalIndent` for `displayMath` in Version 3.0 ............................................. 149
LISTING 560: `noAdditionalIndent` for `displayMath` in Version 3.0 ............................................. 149
Index

— B —
backup files
cycle through, 19
extension settings, 18
maximum number of backup files, 19
number of backup files, 19
overwrite switch, -w, 6
bibliography files, 128

— C —
capturing parenthesis (regex), 33

— D —
delimiters, 101
  advanced settings, 25
  advanced settings of lookForAlignDelims, 24
  ampersand &, 25
  default settings of lookForAlignDelims, 25
  delimiter justification (left or right), 33
  delimiterRegEx, 33, 128
dontMeasure feature, 31
double backslash demonstration, 30
lookForAlignDelims, 25
poly-switches for double back slash, 100
spacing demonstration, 26
  with specialBeginEnd and the -m switch, 101
  within specialBeginEnd blocks, 39

— E —
exit code, 12
  summary, 12

— I —
indentation
  customising indentation per-code block, 42
  customising per-name, 42
  default, 9
defaultIndent description, 24
defaultIndent using -y switch, 9
defaultIndent using YAML file, 14
  maximum indentation, 41
  no additional indent, 42
  no additional indent global, 42
  removing indentation per-code block, 42
  summary, 59

— L —
linebreaks
  summary of poly-switches, 101

— M —
modifying linebreaks
  at the beginning of a code block, 93
  at the end of a code block, 95
  by text wrapping, globally, 65
  by text wrapping, per-code-block, 68
  by using one sentence per line, 83
  surrounding double back slash, 100
  using poly-switches, 92

— P —
poly-switches
  adding blank lines (again!): set to 4, 94, 96
  adding blank lines: set to 3, 94, 95
  adding comments and then line breaks: set to 2, 93, 95
  adding line breaks: set to 1, 93, 95
  blank lines, 98
  conflicting partnering, 105
  conflicting switches, 106, 107
  default values, 92
definition, 92
double backslash, 101
evironment global example, 92
evironment per-code block example, 92
for double back slash (delimiters), 100–103
off by default: set to 0, 92
removing line breaks: set to -1, 97
summary of all poly-switches, 103
  values, 92
  visualisation: ♠, ♥, ♦, ♣, 93

— R —
regular expressions
  a word about, 4
  ampersand alignment, 25, 128
  arguments, 124
  at least one +, 39, 111, 114, 124–126
capturing parenthesis, 33, 128
  character class demonstration, 128
  commands, 124
delimiter alignment for edge or node, 39
delimiter regex at #, 34
delimiter regex at # or \>, 35
delimiter regex at \= or \>, 33
delimiter regex at only \>, 34
delimiterRegEx, 25, 128
dontMeasure feature, cell, 32
dontMeasure feature, row, 33
evironments, 124
fine tuning, 124
horizontal space \h, 39, 42, 91, 114, 115, 124
ifElseFi, 124
keyEqualsValuesBracesBrackets, 124
lowercase alph a-z, 32, 33, 42, 83, 85, 87, 91, 124
modifyLineBreaks, 124
NamedGroupingBracesBrackets, 124
numeric 0-9, 39, 42, 87, 91, 124
replacement switch, -r, 110
substitution field, arraycolsep, 111
substitution field, equation, 113
UnNamedGroupingBracesBrackets, 124
uppercase alph A-Z, 39, 42, 83, 85, 91, 124
using -y switch, 16

— S —
sentences
begin with, 85, 86
end with, 85, 87
follow, 85
indenting, 90
one sentence per line, 83
oneSentencePerLine, 83
removing sentence line breaks, 84
text wrapping, 90

specialBeginEnd
alignment at delimiter, 39
combined with lookForAlignDelims, 39
default settings, 36
delimiterRegEx, 39
delimiterRegEx tweaked, 39
double backslash poly-switch demonstration, 101
IfElsFi example, 37
indentRules example, 54
indentRulesGlobal, 59
introduction, 36
lookForAlignDelims, 101
middle, 37
noAdditionalIndent, 54
noAdditionalIndentGlobal, 59
paragraphsStopAt, 78
poly-switch summary, 103
removeParagraphLineBreaks, 74
searching for special before commands, 37
specifying as verbatim, 39
textWrapOptions, 68
tikz example, 39
update to displaymath V3.0, 142

switches
-c, --cruft definition and details, 10
-d, --onlydefault definition and details, 10
-g, --logfile definition and details, 10
-h, --help definition and details, 6
-k, --check definition and details, 12
-kv, --checkv definition and details, 12
-l in relation to other settings, 16
-l, --local definition and details, 8

— T —
text wrap
quick start, 65
recommended starting point, 81

— V —
verbatim
commands, 20
comparison with -r and -rr switches, 115
environments, 20
in relation to oneSentencePerLine, 89
in relation to paragraphsStopAt, 78
in relation to textWrapOptions, 66
noIndentBlock, 21
poly-switch summary, 103
ry, replacementrespectverb switch, 11, 109
specialBeginEnd, 38
verbatimEnvironments demonstration (-l switch), 16
verbatimEnvironments demonstration (-y switch), 16
within summary of text wrapping, 83

— W —
warning
amalgamate field, 63
be sure to test before use, 2
capture groups, 125
capturing parenthesis for lookForAlignDelims, 33
changing huge (textwrap), 67
editing YAML files, 15
fine tuning, 124
the m switch, 64