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.

\texttt{tl;dr}, a quick start guide is given in Section 1.3 on page 5.
3.1.6 docker users .................................................. 14
3.2 From the command line ......................................... 15
3.3 From arara ....................................................... 21
3.4 Summary of exit codes .......................................... 21

4 indentconfig.yaml, local settings and the -y switch ......... 23
  4.1 indentconfig.yaml and .indentconfig.yaml .................. 23
  4.2 localSettings.yaml and friends ............................. 24
  4.3 The -y|yaml switch ........................................ 25
  4.4 Settings load order ........................................ 25

5 defaultSettings.yaml ............................................ 27
  5.1 Backup and log file preferences ............................. 27
  5.2 Verbatim code blocks ....................................... 29
  5.3 filecontents and preamble .................................. 32
  5.4 Indentation and horizontal space ............................ 33
  5.5 Aligning at delimiters ...................................... 33
    5.5.1 lookForAlignDelims: spacesBeforeAmpersand .......... 38
    5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash ...... 40
    5.5.3 lookForAlignDelims: the dontMeasure feature ......... 41
    5.5.4 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature .. 43
    5.5.5 lookForAlignDelims: lookForChildCodeBlocks .......... 45
    5.5.6 lookForAlignDelims: alignContentAfterDoubleBackSlash ... 45
  5.6 Indent after items, specials and headings .................. 46
  5.7 The code blocks known latexindent.pl ...................... 53
  5.8 noAdditionalIndent and indentRules ......................... 53
    5.8.1 Environments and their arguments ..................... 55
    5.8.2 Environments with items ................................ 62
    5.8.3 Commands with arguments ................................ 63
    5.8.4 ifelsefi code blocks .................................. 65
    5.8.5 specialBeginEnd code blocks ............................ 66
    5.8.6 afterHeading code blocks ................................ 67
    5.8.7 The remaining code blocks ............................... 69
      5.8.7.1 keyEqualsValuesBracesBrackets .................. 69
      5.8.7.2 namedGroupingBracesBrackets .................... 70
      5.8.7.3 UnNamedGroupingBracesBrackets ................. 70
      5.8.7.4 filecontents ...................................... 71
    5.8.8 Summary ................................................ 71
  5.9 Commands and the strings between their arguments .......... 71

6 The -m (modifylinebreaks) switch .............................. 77
  6.1 Text Wrapping ............................................... 79
    6.1.1 Text wrap: overview .................................... 79
    6.1.2 Text wrap: simple examples ............................. 80
    6.1.3 Text wrap: blocksFollow examples ...................... 81
    6.1.4 Text wrap: blocksBeginWith examples ................... 85
    6.1.5 Text wrap: blocksEndBefore examples ................... 87
    6.1.6 Text wrap: trailing comments and spaces ............... 88
    6.1.7 Text wrap: when before/after ........................... 89
    6.1.8 Text wrap: wrapping comments ........................... 91
    6.1.9 Text wrap: huge, tabstop and separator ................ 92
  6.2 oneSentencePerLine: modifying line breaks for sentences .. 93
    6.2.1 oneSentencePerLine: overview .......................... 94
    6.2.2 oneSentencePerLine: sentencesFollow ................... 96
    6.2.3 oneSentencePerLine: sentencesBeginWith ............... 97
    6.2.4 oneSentencePerLine: sentencesEndWith .................. 98
    6.2.5 Features of the oneSentencePerLine routine ............ 99
    6.2.6 oneSentencePerLine: text wrapping and indenting sentences .. 101
CONTENTS

6.2.7 oneSentencePerLine: text wrapping and indenting sentences, when before/after 104
6.2.8 oneSentencePerLine: text wrapping sentences and comments 105
6.3 Poly-switches 105
6.3.1 Poly-switches for environments 106
6.3.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine 106
6.3.1.2 Adding line breaks: EndStartsOnOwnLine and EndFinishesWithLineBreak 108
6.3.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary 110
6.3.1.4 Removing line breaks (poly-switches set to −1) 111
6.3.1.5 About trailing horizontal space 113
6.3.1.6 poly-switch line break removal and blank lines 113
6.3.2 Poly-switches for double backslash 115
6.3.2.1 Double backslash starts on own line 115
6.3.2.2 Double backslash finishes with line break 116
6.3.2.3 Double backslash poly-switches for specialBeginEnd 117
6.3.2.4 Double backslash poly-switches for optional and mandatory arguments 117
6.3.2.5 Double backslash optional square brackets 118
6.3.3 Poly-switches for other code blocks 119
6.3.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches 121
6.3.5 Conflicting poly-switches: sequential code blocks 122
6.3.6 Conflicting poly-switches: nested code blocks 123
7 The -r, -rv and -rr switches 126
7.1 Introduction to replacements 126
7.2 The two types of replacements 127
7.3 Examples of replacements 127
8 The –lines switch 135
9 Fine tuning 141
10 Conclusions and known limitations 149
11 References 150
11.1 perl-related links 150
11.2 conda-related links 150
11.3 VSCode-related links 150
11.4 Other links 150
11.5 Contributors (in chronological order) 151
A Required Perl modules 153
A.1 Module installer script 153
A.2 Manually installing modules 154
A.2.1 Linux 154
A.2.1.1 perlbrew 154
A.2.1.2 Ubuntu/Debian 154
A.2.1.3 Ubuntu: using the texlive from apt-get 154
A.2.1.4 Ubuntu: users without perl 154
A.2.1.5 Arch-based distributions 154
A.2.1.6 Alpine 155
A.2.2 Mac 155
A.2.3 Windows 155
A.3 The GCString switch 156
B Updating the path variable 157
B.1 Add to path for Linux 157
B.2 Add to path for Windows 157
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 [27] I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar [2] 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 11.5 on page 151 for their valued contributions, and thank you to those who report bugs and request features at [34].

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 28) 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 [34] 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 [34].

If you have used any version 2.* of latexindent.pl, there are a few changes to the interface; see appendix L on page 173 and the comments throughout this document for details.

1.3 Quick start
If you’d like to get started with latexindent.pl then simply type

cmh:~$ latexindent.pl myfile.tex

from the command line.
We give an introduction to latexindent.pl using Listing 1; there is no explanation in this section, which is deliberate for a quick start. The rest of the manual is more verbose.

**LISTING 1: quick-start.tex**

```latex
\documentclass{article}
\usepackage[inner=2.5cm,]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.
\end{myenv}
\end{document}
```

Running

```bash
$ latexindent.pl quick-start.tex
```

gives Listing 2.

**LISTING 2: quick-start-default.tex**

```latex
\documentclass{article}
\usepackage[inner=2.5cm,]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.
\end{myenv}
\end{document}
```

**example 1** Running

```bash
$ latexindent.pl -l quick-start1.yaml quick-start.tex
```

gives Listing 3.
1.3 Quick start

LISTING 3: quick-start-mod1.tex
\documentclass{article}
\usepackage[inner=2.5cm, ]{geometry}
\begin{document}
A quick start demonstration for latexindent.pl.
\begin{myenv}
The body of environments and other code blocks receive indentation.
\end{myenv}
\end{document}

See Section 5.4.

example 2 Running

```bash
$ latexindent.pl -l quick-start2.yaml quick-start.tex
```
gives Listing 5.

LISTING 5: quick-start-mod2.tex
\documentclass{article}
\usepackage[inner=2.5cm, ]{geometry}
\begin{document}
A quick start demonstration for latexindent.pl.
\begin{myenv}
\quad The body of environments and other code blocks receive indentation.
\end{myenv}
\end{document}

See Section 5.8.

example 3 Running

```bash
$ latexindent.pl -l quick-start3.yaml quick-start.tex
```
gives Listing 7.
1.3 Quick start

example 4

Running

```
$ latexindent.pl -m -l quick-start4.yaml quick-start.tex
```

gives Listing 9.

example 5

Running

```
$ latexindent.pl -m -l quick-start5.yaml quick-start.tex
```

gives Listing 11.
1.3 Quick start

LISTING 11: quick-start-mod5.tex
\documentclass{article}
\usepackage[inner=2.5cm,]{geometry}
\begin{document}
A quick start demonstration for latexindent.pl.
\begin{myenv}
The body of environments and other code blocks receive indentation.
\end{myenv}
\end{document}

Full details of text wrapping in Section 6.1.

example 6 Running

cmh:$ latexindent.pl -m -l quick-start6.yaml quick-start.tex

gives Listing 13.

LISTING 13: quick-start-mod6.tex
\documentclass{article}
\usepackage[inner=2.5cm,]{geometry}\begin{document}
A quick start demonstration for latexindent.pl.\begin{myenv}
The body of environments and other code blocks receive indentation.
\end{myenv}
\end{document}

This is an example of a poly-switch; full details of poly-switches are covered in Section 6.3.

example 7 Running

cmh:$ latexindent.pl -m -l quick-start7.yaml quick-start.tex

gives Listing 15.
1.3 Quick start

LISTING 15: quick-start-mod7.tex
\documentclass{article}
\usepackage[
inner=2.5cm,\
]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive
indentation.\end{myenv}\end{document}

Full details of \textit{poly-switches} are covered in Section 6.3.

\textbf{example 8} Running

\texttt{cmh:~\$ latexindent.pl -l quick-start8.yaml quick-start.tex}

gives Listing 17; note that the \textit{preamble} has been indented.

LISTING 17: quick-start-mod8.tex
\documentclass{article}
\usepackage[
inner=2.5cm,\
]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.\end{myenv}
\end{document}

See Section 5.3.

\textbf{example 9} Running

\texttt{cmh:~\$ latexindent.pl -l quick-start9.yaml quick-start.tex}

gives Listing 19.
1.4 Required perl modules

If you receive an error message such as that given in Listing 21, then you need to install the missing perl modules.

```
Can't locate File/HomeDir.pm in @INC (@INC contains: /Library/Perl/5.12/darwin-thread-multi-2level /Library/Perl/5.12 /Network/Library/Perl/5.12/darwin-thread-multi-2level /Network/Library/Perl/5.12 /Library/Perl/Updates/5.12.4/darwin-thread-multi-2level /Library/Perl/Updates/5.12.4 /System/Library/Perl/5.12/darwin-thread-multi-2level /System/Library/Perl/5.12/darwin-thread-multi-2level /System/Library/Perl/Extras/5.12/j at helloworld.pl line 10.
BEGIN failed--compilation aborted at helloworld.pl line 10.
```

latexindent.pl ships with a script to help with this process; if you run the following script, you should be prompted to install the appropriate modules.

```
cmh:~$ perl latexindent-module-installer.pl
```

You might also like to see [https://stackoverflow.com/questions/19590042/error-cant-locate-file-homedir-pm-in-inc](https://stackoverflow.com/questions/19590042/error-cant-locate-file-homedir-pm-in-inc), for example, as well as appendix A on page 153.

1.5 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 606. 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:

This type of listing is a .tex file.

This type of listing is a .yaml file; when you see line numbers given (as here) it means that the snippet is taken directly from defaultSettings.yaml, discussed in detail in Section 5 on page 27.
1.6 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 [33].
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 [47]

As you look at Listings 26 to 31, 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 26 to 31 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).

### Listing 26: filecontents1.tex
```latex
\begin{filecontents}{mybib.bib}
@online{strawberryperl,
  title="Strawberry Perl",
  url="http://strawberryperl.com/"}
@online{cmhblog,
  title="A Perl script ...",
  url="...
}
\end{filecontents}
```

### Listing 27: filecontents1.tex default output
```latex
\begin{filecontents}{mybib.bib}
@online{strawberryperl,
  title="Strawberry Perl",
  url="http://strawberryperl.com/"}
@online{cmhblog,
  title="A Perl script ...",
  url="...
}
\end{filecontents}
```

### Listing 28: tikzset.tex
```latex
\tikzset{
  shrink inner sep/.code={
    \pgfkeysgetvalue...
    \pgfkeysgetvalue...
  }
}
```

### Listing 29: tikzset.tex default output
```latex
\tikzset{
  shrink inner sep/.code={
    \pgfkeysgetvalue...
    \pgfkeysgetvalue...
  }
}
```

### Listing 30: pstricks.tex
```latex
\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},\%
  \ldots\%
  }\expandafter...
}
\end{pspicture}
```

### Listing 31: pstricks.tex default output
```latex
\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},\%
  \ldots\%
  }\expandafter...
}
\end{pspicture}
```
SECTION 3

How to use the script

\texttt{latexindent.pl} ships as part of the \TeXLive\ distribution for Linux and Mac users; \texttt{latexindent.exe} ships as part of the \TeXLive\ for Windows users. These files are also available from github [34] should you wish to use them without a \TeX distribution; in this case, you may like to read appendix B on page 157 which details how the \texttt{path} variable can be updated.

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

There are two ways to use \texttt{latexindent.pl}: from the command line, and using arara; we discuss these in Section 3.2 and Section 3.3 respectively. We will discuss how to change the settings and behaviour of the script in Section 5 on page 27.

3.1 Requirements

3.1.1 Perl users

Perl users will need a few standard Perl modules – see appendix A on page 153 for details; in particular, note that a module installer helper script is shipped with \texttt{latexindent.pl}.

3.1.2 Windows users without perl

\texttt{latexindent.pl} ships with \texttt{latexindent.exe} for Windows users, so that you can use the script with or without a Perl distribution.

\texttt{latexindent.exe} is available from [34].

MiKTeX users on Windows may like to see [37] for details of how to use \texttt{latexindent.exe} without a Perl installation.

3.1.3 Ubuntu Linux users without perl

\texttt{latexindent.pl} ships with \texttt{latexindent-linux} for Ubuntu Linux users, so that you can use the script with or without a Perl distribution.

\texttt{latexindent-linux} is available from [34].

3.1.4 macOS users without perl

\texttt{latexindent.pl} ships with \texttt{latexindent-macos} for macOS users, so that you can use the script with or without a Perl distribution.

\texttt{latexindent-macos} is available from [34].

3.1.5 conda users

Users of conda should see the details given in appendix E.

3.1.6 docker users

Users of docker should see the details given in appendix F.
3.2 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**

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

This will output only the version number to the terminal.

- **-vv, --vversion**

```
cmh:~$ latexindent.pl -vv
cmh:~$ latexindent.pl --vversion
```

This will output verbose version details to the terminal, including the location of `latexindent.pl` and `defaultSettings.yaml`.

- **-h, --help**

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

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

```
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.

You can instruct `latexindent.pl` to operate on multiple (batches) of files, for example

```
cmh:~$ latexindent.pl myfile1.tex myfile2.tex
```

Full details are given in appendix C on page 159.

- **-w, --overwrite**

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

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.

- **-wd, --overwriteIfDifferent**
This will overwrite myfile.tex but only if the indented text is different from the original. If the indented text is not different from the original, then myfile.tex will not be overwritten.

All other details from the -w switch are relevant here. If you call latexindent.pl with both the -wd and the -w switch, then the -w switch will be deactivated and the -wd switch takes priority.

-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). The same is true for the -wd switch, and the -o switch takes priority over it.

Note that using -o as above is equivalent to using

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:

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:

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,
3.2 From the command line

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 I on page 173 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
cmh:~$ latexindent.pl myfile.tex -s

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

-t, –trace

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

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, –tttrace

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

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
cmh:~$ latexindent.pl -l=myyaml.yaml myfile.tex
cmh:~$ latexindent.pl -l=myyaml.yaml myfile.tex
cmh:~$ latexindent.pl -l=first.yaml,second.yaml,third.yaml myfile.tex
cmh:~$ latexindent.pl -l=first.yaml,second.yaml,third.yaml myfile.tex
cmh:~$ latexindent.pl myfile.tex -l=first.yaml,second.yaml,third.yaml

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 24) in the current 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 commas) of a YAML file(s) that resides in the same directory as `myfile.tex`; you can use this option 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=~/cmhughes/latexindent/myyaml.yaml myfile.tex
$ latexindent.pl -l=C:\Users\cmhughes\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
$ 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
$ 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=yaml` settings

```bash
$ latexindent.pl myfile.tex -y="defaultIndent:\n\n"
$ latexindent.pl myfile.tex -y="defaultIndent:\n\n,maximumIndentation:\n\n"
$ latexindent.pl myfile.tex -y="indentRules:one: \t\t\t\t"
$ latexindent.pl myfile.tex
  -y='modifyLineBreaks:environments:EndStartsOnOwnLine:3' -m
$ latexindent.pl myfile.tex
  -y='modifyLineBreaks:environments:one:EndStartsOnOwnLine:3' -m
```

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 25.
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 25.

\(-d, \text{–onlydefault}\)

```bash
$ 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.

\(-c, \text{–cruft=<directory>}\)

```bash
$ 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.

\(-g, \text{–logfile=<name of log file>}\)

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

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
$ latexindent.pl -g /dev/null myfile.tex
```

\(-s l, \text{–screenlog}\)

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

\(-m, \text{–modifylinebreaks}\)

```bash
$ latexindent.pl -m myfile.tex
$ 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 77

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

```bash
$ latexindent.pl myfile
```
3.2 From the command line

and in which case, you can specify the order in which extensions are searched for; see Listing 36 on page 27 for full details.

**STDIN**

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

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 myfile.tex, then the above command will output the results of operating upon myfile.tex.

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

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

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

Once you have finished typing your input, you can press

- CTRL+D on Linux
- CTRL+Z followed by ENTER on Windows

to signify that your input has finished. Thanks to [9] for an update to this feature.

\(-r\), \(-\text{replacement}\)

```bash
cmh:~$ latexindent.pl \(-r\) myfile.tex
```

You can call latexindent.pl with the \(-r\) switch to instruct it to perform replacements/substitutions on your file; full details and examples are given in Section 7 on page 126.

\(-rv\), \(-\text{replacementrespectverb}\)

```bash
cmh:~$ latexindent.pl \(-rv\) myfile.tex
```

You can instruct latexindent.pl to perform replacements/substitutions by using the \(-rv\) switch, but will respect verbatim code blocks; full details and examples are given in Section 7 on page 126.

\(-rr\), \(-\text{onlyreplacement}\)

```bash
cmh:~$ latexindent.pl \(-rr\) myfile.tex
```

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 126.

\(-k\), \(-\text{check}\)
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

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

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

```bash
cmh:~$ latexindent.pl -n 5-8 myfile.tex
cmh:~$ latexindent.pl --lines 5-8 myfile.tex
```

The `lines` switch instructs `latexindent.pl` to operate only on specific line ranges within `myfile.tex`. Complete demonstrations are given in Section 8.

### --GCString

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

instructs `latexindent.pl` to load the `Unicode::GCString` module. This should only be necessary if you find that the alignment at ampersand routine (described in Section 5.5) does not work for your language. Further details are given in appendix A.3.

### 3.3 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 [1].

### 3.4 Summary of exit codes

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

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

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

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

<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>, <code>-vversion</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>
SECTION 4

indentconfig.yaml, local settings and the \texttt{-y} switch

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

We focus our discussion on \texttt{indentconfig.yaml}, but there are other options which are detailed in appendix H.

4.1 \texttt{indentconfig.yaml} and \texttt{.indentconfig.yaml}

\texttt{latexindent.pl} will always check your home directory for \texttt{indentconfig.yaml} and \texttt{.indentconfig.yaml} (unless it is called with the \texttt{-d} switch), which is a plain text file you can create that contains the absolute paths for any settings files that you wish \texttt{latexindent.pl} to load. There is no difference between \texttt{indentconfig.yaml} and \texttt{.indentconfig.yaml}, other than the fact that \texttt{.indentconfig.yaml} is a 'hidden' file; thank you to [5] for providing this feature. In what follows, we will use \texttt{indentconfig.yaml}, but it is understood that this could equally represent \texttt{.indentconfig.yaml}. If you have both files in existence then \texttt{indentconfig.yaml} takes priority.

For Mac and Linux users, their home directory is \texttt{/username} while Windows (Vista onwards) is \texttt{C:\Users\username}. Listing 32 shows a sample \texttt{indentconfig.yaml} file.

<table>
<thead>
<tr>
<th>Listing 32: \texttt{indentconfig.yaml} (sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Paths to user settings for \texttt{latexindent.pl}</td>
</tr>
<tr>
<td>#</td>
</tr>
<tr>
<td># Note that the settings will be read in the order you</td>
</tr>
<tr>
<td># specify here- each successive settings file will overwrite</td>
</tr>
<tr>
<td># the variables that you specify</td>
</tr>
<tr>
<td>paths:</td>
</tr>
<tr>
<td>- /home/cmhughes/Documents/yamlfiles/mysettings.yaml</td>
</tr>
<tr>
<td>- /home/cmhughes/folder/othersettings.yaml</td>
</tr>
<tr>
<td>- /some/other/folder/anynameyouwant.yaml</td>
</tr>
<tr>
<td>- C:\Users\chughes\Documents\mysettings.yaml</td>
</tr>
<tr>
<td>- C:\Users\chughes\Desktop\test spaces\more spaces.yaml</td>
</tr>
</tbody>
</table>

Note that the .yaml files you specify in \texttt{indentconfig.yaml} will be loaded in the order in which you write them. Each file doesn't have to have every switch from \texttt{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 \texttt{defaultSettings.yaml} in another directory and call it, for example, \texttt{mysettings.yaml}. Once you have added the path to \texttt{indentconfig.yaml} you can change the switches and add more code-block names to it as you see fit – have a look at Listing 33 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.

\footnote{If you're not sure where to put \texttt{indentconfig.yaml}, don't worry \texttt{latexindent.pl} will tell you in the log file exactly where to put it assuming it doesn't exist already.}
### Listing 33: mysettings.yaml (example)

```yaml
# Default value of indentation
defaultIndent: "\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 34.

#### Listing 34: The encoding option for indentconfig.yaml

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

Thank you to [15] for this contribution; please see appendix J on page 171 and details within [41] 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:

```
cmh:~$ 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.

If you'd prefer to name your `localSettings.yaml` file something different, (say, `mysettings.yaml` as in Listing 33) then you can call `latexindent.pl` using, for example,

---

3Windows users may find that they have to end `.yaml` files with a blank line
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 35 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 6.2 on page 93) and the listings within Listing 369 on page 96, the following settings give the option to have sentences end with a semicolon.

```
cmh:~$ latexindent.pl
   --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;
4.4 Settings load order

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.](image_url)
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

```bash
$ 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.

### Listing 36: fileExtensionPreference

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>fileExtensionPreference:</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>.tex: 1</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>.sty: 2</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>.cls: 3</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>.bib: 4</td>
<td></td>
</tr>
</tbody>
</table>

Calling latexindent.pl myfile with the (default) settings specified in Listing 36 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

**backupExtension:** (extension name)

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.

---

4Throughout 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 `latexindent.pl` (so you don’t want `myfile.bak1`, `myfile.bak2`, etc) and you simply want `myfile.bak` (or whatever you chose `backupExtension` to be) then change `onlyOneBackUp` to 1; the default value of `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 `maxNumberOfBackUps` is 0 which will not prevent backup files being made; in this case, the behaviour will be dictated entirely by `onlyOneBackUp`. The default value of `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 `maxNumberOfBackUps`: 4, and `cycleThroughBackUps` set to 1 then the copy procedure given below would be obeyed.

```bash
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
```

The default value of `cycleThroughBackUps` is 0.

**logFilePreferences**: (fields)

`latexindent.pl` writes information to `indent.log`, some of which can be customized by changing `logFilePreferences`; see Listing 37. If you load your own user settings (see Section 4 on page 23) then `latexindent.pl` will detail them in `indent.log`; you can choose not to have the details logged by switching `showEveryYamlRead` to 0. Once all of your settings have been loaded, you can see the amalgamated settings in the log file by switching `showAmalgamatedSettings` to 1, if you wish.

```
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td>logFilePreferences:</td>
</tr>
<tr>
<td>92</td>
<td>showEveryYamlRead: 1</td>
</tr>
<tr>
<td>93</td>
<td>showAmalgamatedSettings: 0</td>
</tr>
<tr>
<td>94</td>
<td>showDecorationStartCodeBlockTrace: 0</td>
</tr>
<tr>
<td>95</td>
<td>showDecorationFinishCodeBlockTrace: 0</td>
</tr>
<tr>
<td>96</td>
<td>endLogFileWith: '------------'</td>
</tr>
<tr>
<td>97</td>
<td>showGitHubInfoFooter: 1</td>
</tr>
<tr>
<td>98</td>
<td>Dumper:</td>
</tr>
<tr>
<td>99</td>
<td>Terse: 1</td>
</tr>
<tr>
<td>100</td>
<td>Indent: 1</td>
</tr>
<tr>
<td>101</td>
<td>Useqq: 1</td>
</tr>
<tr>
<td>102</td>
<td>Deparse: 1</td>
</tr>
<tr>
<td>103</td>
<td>Quotekeys: 0</td>
</tr>
<tr>
<td>104</td>
<td>Sortkeys: 1</td>
</tr>
<tr>
<td>105</td>
<td>Pair: &quot; =&gt; &quot;</td>
</tr>
</tbody>
</table>
```

When either of the trace modes (see page 17) are active, you will receive detailed information in `indent.log`. You can specify character strings to appear before and after the notification of a found code block using, respectively, `showDecorationStartCodeBlockTrace` and `showDecorationFinishCodeBlockTrace`. A demonstration is given in appendix I on page 170.
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 logfile.

Some of the options for Perl's Dumper module can be specified in Listing 37; see [32] and [31] for more information. These options will mostly be helpful for those calling latexindent.pl with the -tt option described in Section 3.2.

## 5.2 Verbatim code blocks

**verbatimEnvironments:** {fields}

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 38.

<table>
<thead>
<tr>
<th>Listing 38: verbatimEnvironments</th>
<th>Listing 39: verbatimCommands</th>
</tr>
</thead>
<tbody>
<tr>
<td>109 verbatimEnvironments:</td>
<td>115 verbatimCommands:</td>
</tr>
<tr>
<td>110     verbatim: 1</td>
<td>116     verb: 1</td>
</tr>
<tr>
<td>111     lstlisting: 1</td>
<td>117     lstinline: 1</td>
</tr>
<tr>
<td>112     minted: 1</td>
<td></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 [18] for contributing this feature.

**example 10**

For demonstration, then assuming that your file contains the environments latexcode, latexcode*, pythoncode and pythoncode*, then the listings given in Listings 40 and 41 are equivalent.

<table>
<thead>
<tr>
<th>Listing 40: nameAsRegex1.yaml</th>
<th>Listing 41: nameAsRegex2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbatimEnvironments:</td>
<td>verbatimEnvironments:</td>
</tr>
<tr>
<td>latexcode: 1</td>
<td>nameAsRegex:</td>
</tr>
<tr>
<td>latexcode*: 1</td>
<td>name: '\w+code*?'</td>
</tr>
<tr>
<td>pythoncode: 1</td>
<td>lookForThis: 1</td>
</tr>
<tr>
<td>pythoncode*: 1</td>
<td></td>
</tr>
</tbody>
</table>

With reference to Listing 41:

- 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).

**verbatimCommands:** {fields}

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 77).

With reference to Listing 39, 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 [18] for contributing this feature.
example 11  For demonstration, then assuming that your file contains the commands `verbinline`, `myinline` then the listings given in Listings 42 and 43 are equivalent.

<table>
<thead>
<tr>
<th>Listing 42: <code>nameAsRegex3.yaml</code></th>
<th>Listing 43: <code>nameAsRegex4.yaml</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>verbatimCommands:</code></td>
<td><code>verbatimCommands:</code></td>
</tr>
<tr>
<td><code>verbinline: 1</code></td>
<td><code>nameAsRegex:</code></td>
</tr>
<tr>
<td><code>myinline: 1</code></td>
<td><code>name: '\w+inline'</code></td>
</tr>
<tr>
<td></td>
<td><code>lookForThis: 1</code></td>
</tr>
</tbody>
</table>

With reference to Listing 43:

- 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 44.

<table>
<thead>
<tr>
<th>Listing 44: <code>noIndentBlock</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>122 <code>noIndentBlock:</code></td>
</tr>
<tr>
<td>123 <code>noindent: 1</code></td>
</tr>
<tr>
<td>124 <code>cmhtest: 1</code></td>
</tr>
</tbody>
</table>

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 45 for example.

<table>
<thead>
<tr>
<th>Listing 45: <code>noIndentBlock.tex</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>% \begin{noindent}</code></td>
</tr>
<tr>
<td>some before text</td>
</tr>
<tr>
<td>this code</td>
</tr>
<tr>
<td>won't</td>
</tr>
<tr>
<td>be touched</td>
</tr>
<tr>
<td>by <code>latexindent.pl</code>!</td>
</tr>
<tr>
<td>some after text</td>
</tr>
<tr>
<td><code>% \end{noindent}</code></td>
</tr>
</tbody>
</table>

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

example 12  The `noIndentBlock` fields can also be specified in terms of `begin` and `end` fields. We use the code in Listing 46 to demonstrate this feature.

<table>
<thead>
<tr>
<th>Listing 46: <code>noIndentBlock1.tex</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>some before text</td>
</tr>
<tr>
<td>this code</td>
</tr>
<tr>
<td>won't</td>
</tr>
<tr>
<td>be touched</td>
</tr>
<tr>
<td>by <code>latexindent.pl</code>!</td>
</tr>
<tr>
<td>some after text</td>
</tr>
</tbody>
</table>

The settings given in Listings 47 and 48 are equivalent:
5.2 Verbatim code blocks

Upon running the commands

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

then we receive the output given in Listing 50.

```
\begin{verbatim}
some before text
   this code
      won't
   be touched
       by
   latexindent.pl!
some after text
\end{verbatim}
```

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 47. 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).

**example 13** Using Listing 49 demonstrates setting `lookForThis` to 0 (off); running the command

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

gives the output in Listing 51.

```
\begin{verbatim}
some before text
   this code
      won't
   be touched
       by
   latexindent.pl!
some after text
\end{verbatim}
```

We will demonstrate this feature later in the documentation in Listing 572.

You can, optionally, specify the `noIndentBlock` field using the `name` field which takes a regular expression as its argument; thank you to [18] for contributing this feature.

**example 14** For demonstration, then assuming that your file contains the environments `testnoindent`, `testnoindent*` then the listings given in Listings 52 and 53 are equivalent.
With reference to Listing 53:

- 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 54. The behaviour of `latexindent.pl` on these environments is determined by their location (preamble or not), and the value `indentPreamble`, discussed next.

**Listing 54: fileContentsEnvironments**

```yaml
dict:
  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 55, 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 55: lookForPreamble**

```yaml
dict:
  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 56.
5.4 Indentation and horizontal space

**Listing 56: Motivating preambleCommandsBeforeEnvironments**

...  
preheadhook={\begin{mdframed}[style=myframedstyle]},  
postfoothook=\end{mdframed},  
...

---

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 53.

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 57; each of the fields can take the values 0 or 1. See Listings 460 to 462 on page 113 for before and after results. Thanks to [3] for providing this feature.

**Listing 57: removeTrailingWhitespace**

```
removeTrailingWhitespace:  
beforeProcessing: 0  
afterProcessing: 1
```

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 58.

---

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 59). In fact, the fields in lookForAlignDelims can actually take two different forms: the basic version is shown in Listing 59 and the advanced version in Listing 62; we will discuss each in turn.

**Listing 59: lookForAlignDelims (basic)**

```
lookForAlignDelims:  
tabular: 1  
tabularx: 1  
longtable: 1  
array: 1  
matrix: 1  
...
```

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 60 and 61; running the command

```
cmh:~$ latexindent.pl tabular1.tex
```
5.5 Aligning at delimiters

gives the output given in Listing 61.

<table>
<thead>
<tr>
<th>LISTING 60: tabular1.tex</th>
<th>LISTING 61: tabular1.tex default output</th>
</tr>
</thead>
</table>
| `\begin{tabular}{cccc}
1 & 2 & 3 & 4 \\
5 & \& 6 & \\
\end{tabular}` | `\begin{tabular}{cccc}
1 & 2 & 3 & 4 \\
5 & & 6 & \\
\end{tabular}` |

If you find that `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 `noIndentBlock` (see Listing 44 on page 30).

If, for example, you wish to remove the alignment of the `\` within a delimiter-aligned block, then the advanced form of `lookForAlignDelims` shown in Listing 62 is for you.

<table>
<thead>
<tr>
<th>LISTING 62: lookForAlignDelims (advanced)</th>
</tr>
</thead>
<tbody>
<tr>
<td>158 <code>lookForAlignDelims:</code></td>
</tr>
<tr>
<td>159 <code>tabular:</code></td>
</tr>
<tr>
<td>160 <code>delims: 1</code></td>
</tr>
<tr>
<td>161 <code>alignDoubleBackSlash: 1</code></td>
</tr>
<tr>
<td>162 <code>spacesBeforeDoubleBackSlash: 1</code></td>
</tr>
<tr>
<td>163 <code>multiColumnGrouping: 0</code></td>
</tr>
<tr>
<td>164 <code>alignRowsWithoutMaxDelims: 1</code></td>
</tr>
<tr>
<td>165 <code>spacesBeforeAmpersand: 1</code></td>
</tr>
<tr>
<td>166 <code>spacesAfterAmpersand: 1</code></td>
</tr>
<tr>
<td>167 <code>justification: left</code></td>
</tr>
<tr>
<td>168 <code>alignFinalDoubleBackSlash: 0</code></td>
</tr>
<tr>
<td>169 <code>dontMeasure: 0</code></td>
</tr>
<tr>
<td>170 <code>delimiterRegEx: (?&lt;!\)(&amp;)</code></td>
</tr>
<tr>
<td>171 <code>delimiterJustification: left</code></td>
</tr>
<tr>
<td>172 <code>lookForChildCodeBlocks: 1</code></td>
</tr>
<tr>
<td>173 <code>alignContentAfterDoubleBackSlash: 0</code></td>
</tr>
<tr>
<td>174 <code>spacesAfterDoubleBackSlash: 1</code></td>
</tr>
<tr>
<td>175 <code>tabularx:</code></td>
</tr>
<tr>
<td>176 <code>delims: 1</code></td>
</tr>
</tbody>
</table>

Note that you can use a mixture of the basic and advanced form: in Listing 62 `tabular` and `tabularx` are advanced and `longtable` is basic. When using the advanced form, each field should receive at least 1 sub-field, and can (but does not have to) receive any of the following fields:

- `delims`: binary switch (0 or 1) equivalent to simply specifying, for example, `tabular: 1` in the basic version shown in Listing 59. If `delims` is set to 0 then the align at ampersand routine will not be called for this code block (default: 1);
- `alignDoubleBackSlash`: binary switch (0 or 1) to determine if `\` should be aligned (default: 1);
- `spacesBeforeDoubleBackSlash`: optionally, specifies the number (integer \(\geq 0\)) of spaces to be inserted before `\` (default: 1);
- `multiColumnGrouping`: binary switch (0 or 1) that details if `latexindent.pl` should group columns above and below a `\multicolumn` command (default: 0);
- `alignRowsWithoutMaxDelims`: binary switch (0 or 1) that details if rows that do not contain the maximum number of delimiters should be formatted so as to have the ampersands aligned (default: 1);
- `spacesBeforeAmpersand`: optionally specifies the number (integer \(\geq 0\)) of spaces to be placed before ampersands (default: 1);
- `spacesAfterAmpersand`: optionally specifies the number (integer \(\geq 0\)) of spaces to be placed after ampersands (default: 1);
5.5 Aligning at delimiters

- **justification**: optionally specifies the justification of each cell as either *left* or *right* (default: left);
- **alignFinalDoubleBackSlash** optionally specifies if the *final* double backslash 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;
- **alignContentAfterDoubleBackSlash** optionally instructs *latexindent.pl* to align content *after* double backslash (default: 0), discussed in Section 5.5.6;
- **spacesAfterDoubleBackSlash** optionally specifies the number (integer ≥ 0) of spaces to be placed *after* the double back slash when *alignContentAfterDoubleBackSlash* is active; demonstrated in Section 5.5.6.

**example 15** We will explore most of these features using the file *tabular2.tex* in Listing 63 (which contains a *\multicolumn* command), and the YAML files in Listings 64 to 70; we will explore **alignFinalDoubleBackSlash** in Listing 91; the **dontMeasure** feature will be described in Section 5.5.3, and **delimiterRegEx** in Section 5.5.4.

**Listing 63: tabular2.tex**

```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 64: tabular2.yaml**

```yaml
lookForAlignDelims:
  tabular:
    multiColumnGrouping: 1
```

**Listing 65: tabular3.yaml**

```yaml
lookForAlignDelims:
  tabular:
    alignRowsWithoutMaxDelims: 0
```

**Listing 66: tabular4.yaml**

```yaml
lookForAlignDelims:
  tabular:
    spacesBeforeAmpersand: 4
```

**Listing 67: tabular5.yaml**

```yaml
lookForAlignDelims:
  tabular:
    spacesAfterAmpersand: 4
```

**Listing 68: tabular6.yaml**

```yaml
lookForAlignDelims:
  tabular:
    alignDoubleBackSlash: 0
```

**Listing 69: tabular7.yaml**

```yaml
lookForAlignDelims:
  tabular:
    spacesBeforeDoubleBackSlash: 0
```
5.5 Aligning at delimiters

LISTING 70: tabular8.yaml

```
lookForAlignDelims:
  tabular:
    justification: "right"
```

On running the commands

```
cmh:~$ latexindent.pl tabular2.tex
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular3.yaml
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular4.yaml
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular5.yaml
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular6.yaml
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular7.yaml
```

```
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular8.yaml
```

we obtain the respective outputs given in Listings 71 to 78.

LISTING 71: tabular2.tex default output

```
\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 72: tabular2.tex using Listing 64

```
\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 73: tabular2.tex using Listing 65

```
\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

\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}

\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}

\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}

\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}

\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}

\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}

\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}

\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}

Notice in particular:

- in both Listings 71 and 72 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 71 the columns have been aligned at the ampersand;
- in Listing 72 the \multicolumn command has grouped the 2 columns beneath and above it,
because multiColumnGrouping is set to 1 in Listing 64;

- in Listing 73 rows 3 and 6 have not been aligned at the ampersand, because alignRowsWithoutMaxDelims has been set to 0 in Listing 65; however, the \ have still been aligned;

- in Listing 74 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 75 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 76 the \ have not been aligned, because alignDoubleBackSlash is set to 0, otherwise the output is the same as Listing 72;

- in Listing 77 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 72;

- in Listing 78 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 64.

5.5.1 lookForAlignDelims: spacesBeforeAmpersand

The spacesBeforeAmpersand can be specified in a few different ways. The basic form is demonstrated in Listing 66, 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.

example 16 We demonstrate this feature in relation to Listing 79; upon running the following command

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

then we receive the default output given in Listing 80.

```
\begin{aligned}
& a & b, \\
& c & d.
\end{aligned}
```

The settings in Listings 81 to 84 are all equivalent; we have used the not-yet discussed noAdditionalIndent field (see Section 5.8 on page 53) which will assist in the demonstration in what follows.

```
<table>
<thead>
<tr>
<th>Listing 81: sba1.yaml</th>
<th>Listing 82: sba2.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: 1</td>
<td>aligned: 1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Listing 83: sba3.yaml</th>
<th>Listing 84: sba4.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>default: 1</td>
<td>leadingBlankColumn: 1</td>
</tr>
</tbody>
</table>
```

Upon running the following commands
then we receive the (same) output given in Listing 85; we note that there is one space before each ampersand.

**Listing 85: aligned1-mod1.tex**

```latex
\begin{aligned}
& a & b, \\
& c & d.
\end{aligned}
```

We note in particular:

- Listing 81 demonstrates the basic form for `lookForAlignDelims`; in this case, the default values are specified as in Listing 62 on page 34;
- Listing 82 demonstrates the advanced form for `lookForAlignDelims` and specified `spacesBeforeAmpersand`. The default value is 1;
- Listing 83 demonstrates the new advanced way to specify `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 `leadingBlankColumn` has not been specified in Listing 83, and it will inherit the value from `default`;
- Listing 84 demonstrates spaces to be used before amperands for *leading blank columns*. We note that `default` has not been specified, and it will be set to 1 by default.

**Example 17** We can customise the space before the ampersand in the *leading blank column* of Listing 85 by using either of Listings 86 and 87, which are equivalent.

**Listing 86: sba5.yaml**

```yaml
noAdditionalIndent:
  aligned: 1
lookForAlignDelims:
  aligned:
    spacesBeforeAmpersand:
      leadingBlankColumn: 0
```

**Listing 87: sba6.yaml**

```yaml
noAdditionalIndent:
  aligned: 1
lookForAlignDelims:
  aligned:
    spacesBeforeAmpersand:
      leadingBlankColumn: 0
default: 1
```

Upon running

```
cmh:~$ latexindent.pl aligned1.tex -l sba5.yaml
cmh:~$ latexindent.pl aligned1.tex -l sba6.yaml
```

then we receive the (same) output given in Listing 88. We note that the space before the ampersand in the *leading blank column* has been set to 0 by Listing 87.

We can demonstrated this feature further using the settings in Listing 90 which give the output in Listing 89.
5.5 Aligning at delimiters

Listing 88: aligned1-mod5.tex
\begin{aligned}
& a & b, \\
& c & d.
\end{aligned}

Listing 89: aligned1.tex using Listing 90
\begin{aligned}
& a & b, \\
& c & d.
\end{aligned}

Listing 90: sba7.yaml
noAdditionalIndent:
  aligned: 1
lookForAlignDelims:
  aligned:
    spacesBeforeAmpersand:
      leadingBlankColumn: 3
    default: 0

5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash

There may be times when a line of a code block contains more than \\
and in which case, you may want the final double backslash to be aligned.

**example 18** We explore the alignFinalDoubleBackSlash feature by using the file in Listing 91. Upon running the following commands

```bash
cmh:~$ latexindent.pl tabular4.tex -o=+-default

cmh:~$ latexindent.pl tabular4.tex -o=+-FDBS
  -y="lookForAlignDelims:tabular:alignFinalDoubleBackSlash:1"
```

then we receive the respective outputs given in Listing 92 and Listing 93.

We note that in:
- Listing 92, by default, the first set of double backslashes in the first row of the tabular environment have been used for alignment;
- Listing 93, 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 specialBeginEnd on page 47).

**example 19** Assuming that you have a command called \texttt{\matrix} and that it is populated within lookForAlignDelims (which it is, by default), and that you run the command

```bash
cmh:~$ latexindent.pl matrix1.tex
```

then the before-and-after results shown in Listings 94 and 95 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}}...\texttt{\end{tabular}}, then you can use the mark up illustrated in Listing 96; the default output is shown in Listing 97. 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 `lookForAlignDelims`.

<table>
<thead>
<tr>
<th>Listing 96: <code>align-block.tex</code></th>
<th>Listing 97: <code>align-block.tex</code> default output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\begin{tabular} \ 1 &amp; 2 &amp; 3 &amp; 4 \ \ 5 &amp; 6 \ %\end{tabular}</code></td>
<td><code>\begin{tabular} \ 1 &amp; 2 &amp; 3 &amp; 4 \ \ 5 &amp; 6 \ %\end{tabular}</code></td>
</tr>
</tbody>
</table>

With reference to Table 2 on page 54 and the, yet undisussed, fields of `noAdditionalIndent` and `indentRules` (see Section 5.8 on page 53), these comment-marked blocks are considered environments.

### 5.5.3 `lookForAlignDelims`: the `dontMeasure` feature

The `lookForAlignDelims` field can, optionally, receive the `dontMeasure` option which can be specified in a few different ways.

**Example 20**  
We will explore this feature in relation to the code given in Listing 98; the default output is shown in Listing 99.

<table>
<thead>
<tr>
<th>Listing 98: <code>tabular-DM.tex</code></th>
<th>Listing 99: <code>tabular-DM.tex</code> default output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\begin{tabular}{cccc} \ aaaa &amp; bbb &amp; ccc &amp; ddd \ 11 &amp; 2 &amp; 3 &amp; 4 \ 56 &amp; 78 \ %\end{tabular}</code></td>
<td><code>\begin{tabular}{cccc} \ aaaa &amp; bbb &amp; ccc &amp; ddd \ 11 &amp; 2 &amp; 3 &amp; 4 \ 56 &amp; 78 \ %\end{tabular}</code></td>
</tr>
</tbody>
</table>

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 101, we can run the command

```
$ latexindent .pl tabular-DM.tex -l=dontMeasure1.yaml
```
and receive the output given in Listing 100.

<table>
<thead>
<tr>
<th>Listing 100: <code>tabular-DM.tex</code> using Listing 101</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\begin{tabular}{cccc} \ aaaa &amp; bbb &amp; ccc &amp; ddd \ 11 &amp; 2 &amp; 3 &amp; 4 \ 56 &amp; 78 \ %\end{tabular}</code></td>
</tr>
</tbody>
</table>

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

**Example 21**  
The `dontMeasure` field can also be specified in the form demonstrated in Listing 103. On running the following commands,

```
$ latexindent .pl tabular-DM.tex -l=dontMeasure2.yaml
```
we receive the output in Listing 102.
5.5 Aligning at delimiters

We note that in Listing 103 we have specified entries not to be measured, one entry per line.

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

\textbf{Listing 103:} dontMeasure2.yaml

\begin{verbatim}
lookForAlignDelims:
  tabular:
    dontMeasure:
      - aaaaaa
      - bbbbbb
      - ccc
      - dd
\end{verbatim}

We note that in:

\begin{itemize}
  \item Listing 105 we have specified entries not to be measured, each one has a \textit{string} in the \texttt{this} field, together with an optional specification of \texttt{applyTo} as \texttt{cell};
  \item Listing 106 we have specified entries not to be measured as a \textit{regular expression} using the \texttt{regex} field, together with an optional specification of \texttt{applyTo} as \texttt{cell} field, together with an optional specification of \texttt{applyTo} as \texttt{cell}.
\end{itemize}

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

\textbf{example 22}

The \texttt{dontMeasure} field can also be specified in the forms demonstrated in Listing 105 and Listing 106. Upon running the commands

\begin{verbatim}
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure3.yaml
\end{verbatim}

we receive the output given in Listing 104

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

\textbf{Listing 104:} tabular-DM.tex using Listing 105 or Listing 105

\begin{verbatim}
lookForAlignDelims:
  tabular:
    dontMeasure:
      - this: aaaaaa
        applyTo: cell
      - this: bbbbbb
      - ccc
      - dd
\end{verbatim}

\textbf{Listing 105:} dontMeasure3.yaml

\begin{verbatim}
lookForAlignDelims:
  tabular:
    dontMeasure:
      - regex: [a-z]
        applyTo: cell
\end{verbatim}

\textbf{Listing 106:} dontMeasure4.yaml

\begin{itemize}
  \item Listing 105 we have specified entries not to be measured, each one has a \textit{string} in the \texttt{this} field, together with an optional specification of \texttt{applyTo} as \texttt{cell};
  \item Listing 106 we have specified entries not to be measured as a \textit{regular expression} using the \texttt{regex} field, together with an optional specification of \texttt{applyTo} as \texttt{cell} field, together with an optional specification of \texttt{applyTo} as \texttt{cell}.
\end{itemize}

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

\textbf{example 23}

We may also specify the \texttt{applyTo} field as \texttt{row}, a demonstration of which is given in Listing 108; upon running

\begin{verbatim}
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure5.yaml
\end{verbatim}

we receive the output in Listing 107.

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

\textbf{Listing 107:} tabular-DM.tex using Listing 108

\begin{verbatim}
lookForAlignDelims:
  tabular:
    dontMeasure:
      - this: aaaaaa&bbbbb&ccc&dd\n        applyTo: row
\end{verbatim}

\textbf{Listing 108:} dontMeasure5.yaml

\begin{itemize}
  \item Listing 105 we have specified entries not to be measured, each one has a \textit{string} in the \texttt{this} field, together with an optional specification of \texttt{applyTo} as \texttt{cell};
  \item Listing 106 we have specified entries not to be measured as a \textit{regular expression} using the \texttt{regex} field, together with an optional specification of \texttt{applyTo} as \texttt{cell} field, together with an optional specification of \texttt{applyTo} as \texttt{cell}.
\end{itemize}

In both cases, the default value of \texttt{applyTo} is \texttt{cell}, and does not need to be specified.
5.5 Aligning at delimiters

example 24  Finally, the applyTo field can be specified as row, together with a regex expression. For example, for the settings given in Listing 110, upon running

\begin{verbatim}
cmh:~$ latexindent.pl tabular-DM.tex -l=don'tMeasure6.yaml
\end{verbatim}

we receive the output in Listing 109.

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.

**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.

example 25  We demonstrate how to customise this with respect to the code given in Listing 111; the default output from latexindent.pl is given in Listing 112.

\begin{verbatim}
\begin{tabbing}
\begin{tabular}{cccc}
\end{verbatim}

Let’s say that we wish to align the code at either the \= or \>. We employ the settings given in Listing 114 and run the command

\begin{verbatim}
cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx1.yaml
\end{verbatim}

We note that:

- in Listing 113 the code has been aligned, as intended, at both the \= and \>;
in Listing 114 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 `>`. 

**example 26** We can explore `delimiterRegEx` a little further using the settings in Listing 116 and run the command

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

to receive the output given in Listing 115.

**LISTING 115:** tabbing.tex using Listing 116

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

We note that only the `\>` have been aligned.

**example 27** 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 62 on page 34 remain the same; for example, using the settings in Listing 118, and running

```bash
cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx3.yaml
```

to receive the output given in Listing 117.

**LISTING 117:** tabbing.tex using Listing 118

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

**example 28** It is possible that delimiters specified within `delimiterRegEx` can be of different lengths. Consider the file in Listing 119, and associated YAML in Listing 121. Note that the Listing 121 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 120.

**Listing 119:** tabbing1.tex

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

**Listing 120:** tabbing1-mod4.tex

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

**Listing 121:** delimiterRegEx4.yaml

```
lookForAlignDelims:
  tabbing:
    delimiterRegEx: '\(\?:|>)\)'
```
5.5 Aligning at delimiters

example 29 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 123 and running the command

```bash
cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx5.yaml -o=+-mod5
```
gives the output in Listing 122.

LISTING 122: tabbing1-mod5.tex

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

Note that in Listing 122 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.

example 30 Using the settings from Listing 101 on page 41 on the file in Listing 124 and running the command

```bash
cmh:~$ latexindent.pl tabular-DM-1.tex -l=dontMeasure1.yaml -o=+-mod1
```
gives the output in Listing 125.

LISTING 124: tabular-DM-1.tex

```latex
\begin{tabular}{cc}
1 & 2\only<2->{\ \\
3 & 4}
\end{tabular}
```

LISTING 125: tabular-DM-1-mod1.tex

```latex
\begin{tabular}{cc}
1 & 2\only<2->{\ \\
3 & 4}
\end{tabular}
```

We can improve the output from Listing 125 by employing the settings in Listing 127

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

which gives the output in Listing 127.

LISTING 126: tabular-DM-1-mod1a.tex

```latex
\begin{tabular}{cc}
1 & 2\only<2->{\ \\
3 & 4}
\end{tabular}
```

LISTING 127: dontMeasure1a.yaml

```yaml
lookForAlignDelims:
  tabular:
    dontMeasure: largest
    lookForChildCodeBlocks: 0
```

5.5.6 lookForAlignDelims: alignContentAfterDoubleBackSlash

You can instruct latexindent to align content after the double back slash. See also Section 6.3.2 on page 115.

example 31 We consider the file in Listing 128, and the default output given in Listing 129.
### 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 135 and 136.

#### Listing 134: `indentAfterItems` (fields)

```
indentAfterItems:
  itemize: 1
  itemize*: 1
  enumerate: 1
  enumerate*: 1
  description: 1
  description*: 1
  list: 1
```

#### Listing 135: `items1.tex`

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

#### Listing 136: `items1.tex` default output

```
\begin{itemize}
  \item some text here
  \item some more text here
  \item another item
  \item some more text here
  \end{itemize}
```
5.6 Indent after items, specials and headings

---

**itemNames**: (fields)

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 23 for details of how to configure user settings, and Listing 33 on page 24 in particular.)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>252</td>
<td>itemNames:</td>
</tr>
<tr>
<td>253</td>
<td>item: 1</td>
</tr>
<tr>
<td>254</td>
<td>myitem: 1</td>
</tr>
</tbody>
</table>

**specialBeginEnd**: (fields)

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 138 shows the default settings of specialBeginEnd.

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>258</td>
<td>specialBeginEnd:</td>
</tr>
<tr>
<td>259</td>
<td>displayMath:</td>
</tr>
<tr>
<td>260</td>
<td>begin: (?&lt;!)\               # \ but <em>not</em> [</td>
</tr>
</tbody>
</table>
| 261  | end: \]                      # \]
| 262  | lookForThis: 1               |
| 263  | inlineMath:                  |
| 264  | begin: (?<!\$)(?<!\$)\$        # $ but *not* \$ or $$   |
| 265  | end: \$\$                     # $$                        |
| 266  | lookForThis: 1               |
| 267  | displayMathTeX:              |
| 268  | begin: \$\$                  # $$                        |
| 269  | end: \$\$                    # $$                        |
| 270  | lookForThis: 1               |
| 271  | specialBeforeCommand: 0      |

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 24); indeed, you might like to set up your own special begin and end statements.

**example 33**

A demonstration of the before-and after results are shown in Listings 139 and 140; explicitly, running the command

```
$ cmh:~$ latexindent.pl special1.tex -o=-default
```

gives the output given in Listing 140.

<table>
<thead>
<tr>
<th>Listing 139: special1.tex before</th>
</tr>
</thead>
<tbody>
<tr>
<td>The function $f$ has formula $f(x)=x^2$.</td>
</tr>
<tr>
<td>If you like splitting dollars, $g(x)=f(2x)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 140: special1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>The function $f$ has formula $f(x)=x^2$.</td>
</tr>
<tr>
<td>If you like splitting dollars, $g(x)=f(2x)$</td>
</tr>
</tbody>
</table>
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.

**example 34** For example, consider the file shown in Listing 141.

```
\begin{equation}
\left[ \sqrt{a+b} \right]
\end{equation}
```

Now consider the YAML files shown in Listings 142 and 143:

```
# Listing 142: specialsLeftRight.yaml
specialBeginEnd:
  leftRightSquare:
    begin: '\left['
    end: '\right]'
  lookForThis: 1

# Listing 143: specialBeforeCommand.yaml
specialBeginEnd:
  specialBeforeCommand: 1
```

Upon running the following commands:

```
cmh:\$ latexindent.pl specialLR.tex -l=specialsLeftRight.yaml
cmh:\$ latexindent.pl specialLR.tex -l=specialsLeftRight.yaml,specialBeforeCommand.yaml
```

we receive the respective outputs in Listings 144 and 145.

```
# Listing 144: specialLR.tex using Listing 142
\begin{equation}
\left[ \sqrt{a+b} \right]
\end{equation}
```

```
# Listing 145: specialLR.tex using Listings 142 and 143
\begin{equation}
\left[ \sqrt{a+b} \right]
\end{equation}
```

Notice that in:

- Listing 144 the `\left` has been treated as a `command`, with one optional argument;
- Listing 145 the `specialBeginEnd` pattern in Listing 142 has been obeyed because Listing 143 specifies that the `specialBeginEnd` should be sought before commands.

You can, optionally, specify the `middle` field for anything that you specify in `specialBeginEnd`.

**example 35** For example, let’s consider the `.tex` file in Listing 146.
5.6 Indent after items, specials and headings

Upon saving the YAML settings in Listings 148 and 150 and running the commands

```bash
cmh:~$ latexindent.pl special2.tex -l=middle

cmh:~$ latexindent.pl special2.tex -l=middle1
```

then we obtain the output given in Listings 147 and 149.

We note that:

- in Listing 147 the bodies of each of the `ElseIf` statements have been indented appropriately;
- the `Else` statement has *not* been indented appropriately in Listing 147 – read on!
- we have specified multiple settings for the `middle` field using the syntax demonstrated in Listing 150 so that the body of the `Else` statement has been indented appropriately in Listing 149.

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

---

**Listing 146: special2.tex**

```
\If
something 0
\ElseIf
  something 1
\ElseIf
  something 2
\ElseIf
  something 3
  \Else
  something 4
\EndIf
```

**Listing 147: special2.tex using Listing 148**

```
\If
  something 0
\ElseIf
  something 1
\ElseIf
  something 2
\ElseIf
  something 3
  \Else
  something 4
\EndIf
```

**Listing 148: middle.yaml**

```
specialBeginEnd:
  If:
    begin: '\If'
    middle: '\ElseIf'
    end: '\EndIf'
  lookForThis: 1
```

**Listing 149: special2.tex using Listing 150**

```
\If
  something 0
\ElseIf
  something 1
\ElseIf
  something 2
\ElseIf
  something 3
  \Else
  something 4
\EndIf
```

**Listing 150: middle1.yaml**

```
specialBeginEnd:
  If:
    begin: '\If'
    middle:
      - '\ElseIf'
      - '\Else'
    end: '\EndIf'
  lookForThis: 1
```
example 36 For example, beginning with the code in Listing 151 and the YAML in Listing 152, and running

```
$ latexindent \special3.tex -l=special-verb1
```

then the output in Listing 151 is unchanged.

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

We can combine the `specialBeginEnd` with the `lookForAlignDelims` feature.

example 37 We begin with the code in Listing 153.

```
\begin{tikzpicture}
\path (A) edge node {0,1,L} (B)
edge node {1,1,R} (C)
\path (B) edge [loop above] node {1,1,L} (B)
edge node {0,1,L} (C)
\path (C) edge node {0,1,L} (D)
\path (D) edge [loop below] node {1,1,R} (D)
\path (E) edge [bend left] node {1,0,R} (A)
\path (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 155 and run the command

```
$ latexindent \special-align.tex -l edge-node1.yaml -o=+-mod1
```

to receive the output in Listing 154.

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

The output in Listing 154 is not quite ideal. We can tweak the settings within Listing 155 in order to improve the output; in particular, we employ the code in Listing 157 and run the command

```
$ latexindent \special-align.tex -l edge-node2.yaml -o=+-mod2
```
to receive the output in Listing 156.

**LISTING 156: special-align.tex using Listing 157**

```
\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,L} (A)
  (E) edge [bend left] node {1,0,R} (A);
\end{tikzpicture}
```

The `lookForThis` field can be considered optional; by default, it is assumed to be 1, which is demonstrated in Listing 157.

**LISTING 157: edge-node2.yaml**

```
specialBeginEnd:
  path:
    begin: '\path'
    end: ';;'
  specialBeforeCommand: 1

lookForAlignDelims:
  path:
    delimiterRegEx: '(edge|node\h*\{[0-9,A-Z]+\})'
```

This field enables the user to specify indentation rules that take effect after heading commands such as \part, \chapter, \section, \subsection*, or indeed any user-specified command written in this field.\(^\text{5}\)

**LISTING 158: indentAfterHeadings**

```
indentAfterHeadings:
  part:
    indentAfterThisHeading: 0
    level: 1
  chapter:
    indentAfterThisHeading: 0
    level: 2
  section:
    indentAfterThisHeading: 0
    level: 3
```

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 53); 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).

**example 38** For example, assuming that you have the code in Listing 160 saved into `headings1.yaml`, and that you have the text from Listing 159 saved into `headings1.tex`.

\(^\text{5}\)There is a slight difference in interface for this field when comparing Version 2.2 to Version 3.0; see appendix I on page 173 for details.
5.6 Indent after items, specials and headings

<table>
<thead>
<tr>
<th>Listing 159: headings1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>subsection text</td>
</tr>
<tr>
<td>subsection text</td>
</tr>
<tr>
<td>\paragraph{paragraph title}</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
<tr>
<td>\paragraph{paragraph title}</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
</tbody>
</table>

If you run the command

```
cmh:~$ latexindent.pl headings1.tex -l=headings1.yaml
```
then you should receive the output given in Listing 161.

<table>
<thead>
<tr>
<th>Listing 161: headings1.tex using Listing 160</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>\hphantom{} subsection text</td>
</tr>
<tr>
<td>\hphantom{} subsection text</td>
</tr>
<tr>
<td>\hphantom{} \paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
<tr>
<td>\hphantom{} \paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
</tbody>
</table>

Now say that you modify the YAML from Listing 160 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 162; notice that the paragraph and subsection are at the same indentation level.

<table>
<thead>
<tr>
<th>Listing 162: headings1.tex second modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>\hphantom{} subsection text</td>
</tr>
<tr>
<td>\hphantom{} subsection text</td>
</tr>
<tr>
<td>\hphantom{} \paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
<tr>
<td>\hphantom{} \paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
<tr>
<td>\hphantom{} paragraph text</td>
</tr>
</tbody>
</table>

maximumIndentation: \textit{(horizontal space)}

You can control the maximum indentation given to your file by specifying the `maximumIndentation` field as horizontal space (but not including tabs). This feature uses the `Text::Tabs` module \[45\], and is off by default.

\textbf{example 39} For example, consider the example shown in Listing 163 together with the default output shown in Listing 164.
5.7 The code blocks known latexindent.pl

As of Version 3.0, latexindent.pl processes documents using code blocks; each of these are shown in Table 2.

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 141.

5.8 noAdditionalIndent and indentRules

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

1. noAdditionalIndent for the name of the current (thing);

2. indentRules for the name of the current (thing);

example 40

Now say that, for example, you have the max-indentation1.yaml from Listing 166 and that you run the following command:

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

You should receive the output shown in Listing 165.

Comparing the output in Listings 164 and 165 we notice that the (default) tabs of indentation have been replaced by a single space.

In general, when using the maximumIndentation feature, any leading tabs will be replaced by equivalent spaces except, of course, those found in verbatimEnvironments (see Listing 38 on page 29) or noIndentBlock (see Listing 44 on page 30).
### Table 2: Code blocks known to latexindent.pl

<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-Z0*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-Z0*0-9_\</td>
<td>\mycommand{arguments}</td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>a-zA-Z0*0-9_/\h{}:#-</td>
<td>my key/.style={arguments}</td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>0-9.,a-zA-Z0*&lt;&gt;</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>
<tr>
<td>ifElseFi</td>
<td>@a-zA-Z but must begin with either \if of @if</td>
<td>\ifnum... ... \else ... \fi</td>
</tr>
<tr>
<td>items</td>
<td>User specified, see Listings 134 and 137 on page 46 and on page 47</td>
<td>\begin{enumerate} ... \end{enumerate}</td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>User specified, see Listing 138 on page 47</td>
<td>[ ... ]</td>
</tr>
<tr>
<td>afterHeading</td>
<td>User specified, see Listing 158 on page 51</td>
<td>\begin{chapter}{title} ... \section{title}</td>
</tr>
<tr>
<td>filecontents</td>
<td>User specified, see Listing 54 on page 32</td>
<td>\begin{filecontents} ... \end{filecontents}</td>
</tr>
</tbody>
</table>


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 167.

**Listing 167: myenv.tex**

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

**noAdditionalIndent**: \{fields\}

**example 41** If we do not wish `myenv` to receive any additional indentation, we have a few choices available to us, as demonstrated in Listings 168 and 169.

**Listing 168:**

```
myenv-noAdd1.yaml
```

```
noAdditionalIndent:
  myenv: 1
```

**Listing 169:**

```
myenv-noAdd2.yaml
```

```
noAdditionalIndent:
  myenv:
    body: 1
```

On applying either of the following commands,
Upon changing the YAML files to those shown in Listings 171 and 172, and running either

\texttt{cmh:\$ latexindent.pl myenv.tex -l myenv-noAdd3.yaml}
\texttt{cmh:\$ latexindent.pl myenv.tex -l myenv-noAdd4.yaml}

we obtain the output given in Listing 173.

\textbf{Listing 173: myenv.tex output (using either Listing 171 or Listing 172)}

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

\textbf{Example 43} Let's now allow \texttt{myenv} to have some optional and mandatory arguments, as in Listing 174.

\textbf{Listing 174: myenv-args.tex}

\begin{verbatim}
\begin{outer}
  \begin{myenv}[%  
    \begin{myenv}[%
      \begin{myenv}[%  
        body of environment
        body of environment
      \end{myenv}%
    \end{myenv}]
    \end{myenv}
  \end{outer}
\end{verbatim}

Upon running
we obtain the output shown in Listing 175; 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 168), then all parts of the environment (body, optional and mandatory arguments) are assumed to want no additional indent.

\begin{outer}
\begin{myenv}
  \{ mandatory argument text \\
  \{ mandatory argument text \\
  body of environment \\
  body of environment \\
  body of environment
\end{myenv}
\end{outer}

\textbf{example 44} We may customise noAdditionalIndent for optional and mandatory arguments of the myenv environment, as shown in, for example, Listings 176 and 177.

Upon running

\begin{verbatim}
cmh:~$ latexindent.pl -l=myenv-noAdd5.yaml myenv-args.tex
\end{verbatim}

we obtain the respective outputs given in Listings 178 and 179. Note that in Listing 178 the text for the optional argument has not received any additional indentation, and that in Listing 179 the mandatory argument has not received any additional indentation; in both cases, the body has not received any additional indentation.

\begin{verbatim}
snoAdditionalIndent: myenv: 
  body: 0 
  optionalArguments: 1 
  mandatoryArguments: 0
\end{verbatim}

\begin{verbatim}
snoAdditionalIndent: myenv: 
  body: 0 
  optionalArguments: 0 
  mandatoryArguments: 1
\end{verbatim}

\\indentRules: (fields)
example 45  We may also specify indentation rules for environment code blocks using the \texttt{indentRules} field; see, for example, Listings 180 and 181.

\begin{Verbatim}
\texttt{myenv-rules1.yaml}
\end{Verbatim}
\begin{Verbatim}
\texttt{myenv-rules2.yaml}
\end{Verbatim}

On applying either of the following commands,

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

we obtain the output given in Listing 182; note in particular that the environment \texttt{myenv} has received one tab (from the \texttt{outer} environment) plus three spaces from Listing 180 or 181.

\begin{Verbatim}
\texttt{myenv.tex}
\end{Verbatim}

\begin{Verbatim}
\texttt{myenv-args.tex}
\end{Verbatim}

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

example 46  Returning to the example in Listing 174 that contains optional and mandatory arguments. Upon using Listing 180 as in

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

we obtain the output in Listing 183; note that the body, optional argument and mandatory argument of \texttt{myenv} have \textit{all} received the same customised indentation.

\begin{Verbatim}
\texttt{myenv-args.tex}
\end{Verbatim}

example 47  You can specify different indentation rules for the different features using, for example, Listings 184 and 185
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 186 and 187.

```
\begin{outer}
  \begin{myenv}[
    \hspace{3}\hspace{optional\hspace{argument\hspace{text}}
    \hspace{3}\hspace{optional\hspace{argument\hspace{text}}
    \hspace{3}\hspace{\hspace{mandatory\hspace{argument\hspace{text}}
    \hspace{3}\hspace{\hspace{argument\hspace{text}}
    \hspace{3}\hspace{body\hspace{of\hspace{environment\hspace{}}
    \hspace{3}\hspace{body\hspace{of\hspace{environment\hspace{}}
    \hspace{3}\hspace{body\hspace{of\hspace{environment\hspace{}}
  \end{myenv}
\end{outer}
```

Note that in Listing 186, 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 187, 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.

```
\begin{outer}
  \begin{myenv}[
    \hspace{3}\hspace{optional\hspace{argument\hspace{text}}
    \hspace{3}\hspace{optional\hspace{argument\hspace{text}}
    \hspace{3}\hspace{\hspace{mandatory\hspace{argument\hspace{text}}
    \hspace{3}\hspace{\hspace{argument\hspace{text}}
    \hspace{3}\hspace{body\hspace{of\hspace{environment\hspace{}}
    \hspace{3}\hspace{body\hspace{of\hspace{environment\hspace{}}
    \hspace{3}\hspace{body\hspace{of\hspace{environment\hspace{}}
  \end{myenv}
\end{outer}
```

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 188).

```
\begin{listing}
  \begin{noAdditionalIndentGlobal}
    \hspace{3}\hspace{environments: 0 # 0/1}
  \end{noAdditionalIndentGlobal}
\end{listing}
```

**Example 48** Let's say that you change the value of environments to 1 in Listing 188, and that you run

```
cmh:~$ latexindent.pl myenv-args.tex -l env-noAdditionalGlobal.yaml
cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-noAdditionalGlobal.yaml
```

The respective output from these two commands are in Listings 189 and 190; in Listing 189 notice that both environments receive no additional indentation but that the arguments of myenv still do receive indentation. In Listing 190 notice that the outer environment does not receive additional indentation, but because of the settings from myenv-rules1.yaml (in Listing 180 on the previous page), the myenv environment still does receive indentation.
In fact, `noAdditionalIndentGlobal` also contains keys that control the indentation of optional and mandatory arguments; on referencing Listings 191 and 192, we may run the commands

```
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
```

which produces the respective outputs given in Listings 193 and 194. Notice that in Listing 193 the optional argument has not received any additional indentation, and in Listing 194 the mandatory argument has not received any additional indentation.

The final check that `latexindent.pl` will make is to look for `indentRulesGlobal` as detailed in Listing 195.
example 50  If you change the environments field to anything involving horizontal space, say " ", and then run the following commands

```
cmh:~$ latexindent.pl myenv-args.tex -l env-indentRules.yaml
cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-indentRules.yaml
```

then the respective output is shown in Listings 196 and 197. Note that in Listing 196, both the environment blocks have received a single-space indentation, whereas in Listing 197 the outer environment has received single-space indentation (specified by indentRulesGlobal), but myenv has received " ", as specified by the particular indentRules for myenv Listing 180 on page 58.

| Listing 196: myenv-args.tex using
<table>
<thead>
<tr>
<th>Listing 195</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer}</td>
</tr>
<tr>
<td>\begin{myenv}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\end{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{mandatory_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{mandatory_argument_text}[^%</td>
</tr>
<tr>
<td>\end{mandatory_argument_text}[^%</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
</tr>
</tbody>
</table>

| Listing 197: myenv-args.tex using
Listings 180 and 195 |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer}</td>
</tr>
<tr>
<td>\begin{myenv}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\end{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\end{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
</tr>
</tbody>
</table>

example 51  You can specify indentRulesGlobal for both optional and mandatory arguments, as detailed in Listings 198 and 199

<table>
<thead>
<tr>
<th>Listing 198: opt-args-indent-rules-glob.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentRulesGlobal:</td>
</tr>
<tr>
<td>optionalArguments: &quot;\t\t&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 199: mand-args-indent-rules-glob.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentRulesGlobal:</td>
</tr>
<tr>
<td>mandatoryArguments: &quot;\t\t&quot;</td>
</tr>
</tbody>
</table>

Upon running the following commands

```
cmh:~$ latexindent.pl myenv-args.tex -local opt-args-indent-rules-glob.yaml
cmh:~$ latexindent.pl myenv-args.tex -local mand-args-indent-rules-glob.yaml
```

we obtain the respective outputs in Listings 200 and 201. Note that the optional argument in Listing 200 has received two tabs worth of indentation, while the mandatory argument has done so in Listing 201.

| Listing 200: myenv-args.tex using
Listings 198 |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer}</td>
</tr>
<tr>
<td>\begin{myenv}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\end{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{mandatory_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{mandatory_argument_text}[^%</td>
</tr>
<tr>
<td>\end{mandatory_argument_text}[^%</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
</tr>
</tbody>
</table>

| Listing 201: myenv-args.tex using
Listings 199 |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer}</td>
</tr>
<tr>
<td>\begin{myenv}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\end{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\begin{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\end{optional_argument_text}[^%</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\body_of_environment</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
</tr>
</tbody>
</table>
5.8.2 Environments with items

With reference to Listings 134 and 137 on page 46 and on page 47, some commands may contain \texttt{item} commands; for the purposes of this discussion, we will use the code from Listing 135 on page 46.

Assuming that you’ve populated \texttt{itemNames} with the name of your \texttt{item}, you can put the item name into \texttt{noAdditionalIndent} as in Listing 202, although a more efficient approach may be to change the relevant field in \texttt{itemNames} to 0.

\textbf{example 52} Similarly, you can customise the indentation that your \texttt{item} receives using \texttt{indentRules}, as in Listing 203

\begin{verbatim}
LISTING 202: item-noAdd1.yaml

\begin{verbatim}
noAdditionalIndent:
  item: 1
# itemNames:
# item: 0
end{verbatim}

\end{verbatim}

Upon running the following commands

\begin{verbatim}
cmh:~$ latexindent.pl items1.tex -local item-noAdd1.yaml
cmh:~$ latexindent.pl items1.tex -local item-rules1.yaml
\end{verbatim}

the respective outputs are given in Listings 204 and 205; note that in Listing 204 that the text after each \texttt{item} has not received any additional indentation, and in Listing 205, the text after each \texttt{item} has received a single space of indentation, specified by Listing 203.

\begin{verbatim}
LISTING 204: items1.tex using Listing 202
\begin{itemize}
\item some text here
  some more text here
  some more text here
\item another item
  some more text here
\end{itemize}

LISTING 205: items1.tex using Listing 203
\begin{itemize}
  ___\item some, text, here
  ______some, more, text, here
  ______some, more, text, here
  ___\item another, item
  ______some, more, text, here
\end{itemize}
\end{verbatim}

\begin{verbatim}
LISTING 206: items-noAdditionalGlobal.yaml

\begin{verbatim}
indentRules:
  item: 0
noAdditionalIndentGlobal:
  items: 1
end{verbatim}

\end{verbatim}

\begin{verbatim}
LISTING 207: items-indentRulesGlobal.yaml

\begin{verbatim}
indentRules:
  item: 0
indentRulesGlobal:
  items: " "
end{verbatim}

\end{verbatim}

\begin{verbatim}
Upon running the following commands,

\begin{verbatim}
cmh:~$ latexindent.pl items1.tex -local items-noAdditionalGlobal.yaml
cmh:~$ latexindent.pl items1.tex -local items-indentRulesGlobal.yaml
\end{verbatim}

the respective outputs from Listings 204 and 205 are obtained; note, however, that all such \texttt{item} commands without their own individual \texttt{noAdditionalIndent} or \texttt{indentRules} settings would behave as in these listings.
5.8.3 Commands with arguments

example 54  Let’s begin with the simple example in Listing 208; when latexindent.pl operates on this file, the default output is shown in Listing 209. 

Listing 208: mycommand.tex
\mycommand
{mand arg text
 mand arg text}
[
opt arg text
 opt arg text]

Listing 209: mycommand.tex default output
\mycommand
{mand arg text
 mand arg text}
[
opt arg text
 opt arg text]

As in the environment-based case (see Listings 168 and 169 on page 55) we may specify noAdditionalIndent either in ‘scalar’ form, or in ‘field’ form, as shown in Listings 210 and 211.

Listing 210: mycommand-noAdd1.yaml
noAdditionalIndent:
mycommand: 1

Listing 211: mycommand-noAdd2.yaml
noAdditionalIndent:
mycommand:
body: 1

After running the following commands,

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 212 and 213.

Listing 212: mycommand.tex using Listing 210
\mycommand
{mand arg text
 mand arg text}
[
opt arg text
 opt arg text]

Listing 213: mycommand.tex using Listing 211
\mycommand
{mand arg text
 mand arg text}
[
opt arg text
 opt arg text]

Note that in Listing 212 that the ‘body’, optional argument and mandatory argument have all received no additional indentation, while in Listing 213, 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.

*The command code blocks have quite a few subtleties, described in Section 5.9 on page 71.

example 55  We may further customise noAdditionalIndent for mycommand as we did in Listings 176 and 177 on page 57; explicit examples are given in Listings 214 and 215.
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 216 and 217.

**example 56**

Attentive readers will note that the body of `mycommand` in both Listings 216 and 217 has received no additional indent, even though `body` is explicitly set to 0 in both Listings 214 and 215. This is because, by default, `noAdditionalIndentGlobal` for commands is set to 1 by default; this can be easily fixed as in Listings 218 and 219.

After running the following commands,

```
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd5.yaml
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd6.yaml
```

we receive the respective output given in Listings 220 and 221.
Both indentRules and indentRulesGlobal can be adjusted as they were for environment code blocks, as in Listings 184 and 185 on page 59 and Listings 195, 198 and 199 on pages 60–61.

5.8.4 ifelsefi code blocks

example 57  Let's use the simple example shown in Listing 222; when latexindent.pl operates on this file, the output as in Listing 223; note that the body of each of the \if statements have been indented, and that the \else statement has been accounted for correctly.

<table>
<thead>
<tr>
<th>Listing 222: ifelsefil.tex</th>
<th>Listing 223: ifelsefil.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ifodd\radius</td>
<td>\ifodd\radius</td>
</tr>
<tr>
<td>\ifnum\radius&lt;14</td>
<td>\ifnum\radius&lt;14</td>
</tr>
<tr>
<td>\pgfmathparse{100-(\radius)*4};</td>
<td>\pgfmathparse{100-(\radius)*4};</td>
</tr>
<tr>
<td>\else</td>
<td>\else</td>
</tr>
<tr>
<td>\pgfmathparse{200-(\radius)*3};</td>
<td>\pgfmathparse{200-(\radius)*3};</td>
</tr>
<tr>
<td>\fi\fi</td>
<td>\fi\fi</td>
</tr>
</tbody>
</table>

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 224 and 225.

<table>
<thead>
<tr>
<th>Listing 224: ifnum-noAdd.yaml</th>
<th>Listing 225: ifnum-indent-rules.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>indentRules:</td>
</tr>
<tr>
<td>ifnum: 1</td>
<td>ifnum: &quot; &quot;</td>
</tr>
</tbody>
</table>

After running the following commands,

```
cmh:~$ latexindent.pl ifelsefil.tex -local ifnum-noAdd.yaml
cmh:~$ latexindent.pl ifelsefil.tex -l ifnum-indent-rules.yaml
```

we receive the respective output given in Listings 226 and 227; note that in Listing 226, the ifnum code block has not received any additional indentation, while in Listing 227, the ifnum code block has received one tab and two spaces of indentation.

<table>
<thead>
<tr>
<th>Listing 226: ifelsefil.tex using Listing 224</th>
<th>Listing 227: ifelsefil.tex using Listing 225</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ifodd\radius \ifnum\radius&lt;14 \pgfmathparse{100-(\radius)*4}; \else \pgfmathparse{200-(\radius)*3}; \fi\fi</td>
<td>\ifodd\radius \ifnum\radius&lt;14 \pgfmathparse{100-(\radius)*4}; \else \pgfmathparse{200-(\radius)*3}; \fi\fi</td>
</tr>
</tbody>
</table>

example 58  We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 228 and 229.

<table>
<thead>
<tr>
<th>Listing 228: ifelsefi-noAdd-glob.yaml</th>
<th>Listing 229: ifelsefi-indent-rules-global.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndentGlobal:</td>
<td>indentRulesGlobal:</td>
</tr>
<tr>
<td>ifElseFi: 1</td>
<td>ifElseFi: &quot; &quot;</td>
</tr>
</tbody>
</table>

Upon running the following commands
5.8 noAdditionalIndent and indentRules

we receive the outputs in Listings 230 and 231; notice that in Listing 230 neither of the `ifelsefi` code blocks have received indentation, while in Listing 231 both code blocks have received a single space of indentation.

---

**example 59**

We can further explore the treatment of `ifElseFi` code blocks in Listing 232, and the associated default output given in Listing 233; 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 139 on page 47 which has default output shown in Listing 140 on page 47.

---

**example 60**

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 234 and 235.

---

After running the following commands,

```bash
$ latexindent.pl special1.tex -local displayMath-noAdd.yaml
$ latexindent.pl special1.tex -l displayMath-indent-rules.yaml
```

we receive the respective output given in Listings 236 and 237; note that in Listing 236, the `displayMath` code block has *not* received any additional indentation, while in Listing 237, the `displayMath` code block has received three tabs worth of indentation.
The function \( f \) has formula
\[
\begin{align*}
f(x) &= x^2. \\
\end{align*}
\]
If you like splitting dollars,
\[
\begin{align*}
g(x) &= f(2x) \\
\end{align*}
\]

The function \( f \) has formula
\[
\begin{align*}
f(x) &= x^2. \\
\end{align*}
\]
If you like splitting dollars,
\[
\begin{align*}
g(x) &= f(2x) \\
\end{align*}
\]

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 238 and 239.

Upon running the following commands

```
$ latexindent.pl special1.tex -local special-noAdd-glob.yaml
$ latexindent.pl special1.tex -l special-indent-rules-global.yaml
```

we receive the outputs in Listings 240 and 241; notice that in Listing 240 neither of the special code blocks have received indentation, while in Listing 241 both code blocks have received a single space of indentation.

The function \( f \) has formula
\[
\begin{align*}
f(x) &= x^2. \\
\end{align*}
\]
If you like splitting dollars,
\[
\begin{align*}
g(x) &= f(2x) \\
\end{align*}
\]


**5.8.6 afterHeading code blocks**

Let's use the example Listing 242 for demonstration throughout this Section. As discussed on page 52, by default latexindent.pl will not add indentation after headings.

```
$ latexindent.pl headings2.tex -l headings3.yaml
```

we obtain the output in Listing 243. Note that the argument of paragraph has received (default) indentation, and that the body after the heading statement has received (default) indentation.
If we specify `noAdditionalIndent` as in Listing 246 and run the command

```
cmh:~$ latexindent.pl headings2.tex -l headings4.yaml
```

then we receive the output in Listing 245. 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.

**example 63**  
Similarly, if we specify `indentRules` as in Listing 248 and run analogous commands to those above, we receive the output in Listing 247; note that the body, mandatory argument and content after the heading of paragraph have all received three tabs worth of indentation.

**example 64**  
We may, instead, specify `noAdditionalIndent` in ‘field’ form, as in Listing 250 which gives the output in Listing 249.

**example 65**  
Analogously, we may specify `indentRules` as in Listing 252 which gives the output in Listing 251; 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.
5.8.7 The remaining code blocks

Referencing the different types of code blocks in Table 2 on page 54, 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 63, 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 54;
- 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 556 on page 141

example 67 An example is shown in Listing 257, with the default output given in Listing 258.
5.8 noAdditionalIndent and indentRules

\begin{Verbatim}
\texttt{\pgfkeys{/tikz/.cd,}
  \texttt{start coordinate/.initial={0,}}
  \texttt{\textbackslash vertfactor},
\}
\end{Verbatim}

In Listing 258, note that the maximum indentation is three tabs, and these come from:

- the \texttt{\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 53.

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 54;
- 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 556 on page 141.

\begin{Verbatim}
\texttt{\textbackslash coordinate}
\texttt{\texttt{\textbackslash coordinate}}
\texttt{child\[grow=down\]}
\texttt{edge from parent \texttt{\[antiparticle\]}}
\texttt{node \texttt{\[above=3pt\]} \texttt{$\texttt{\$\$C\$\$}$}}
\}
\end{Verbatim}

In particular, latexindent.pl considers child, parent and node all to be namedGroupingBracesBrackets*. Referencing Listing 260, 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 53.

\*You may like to verify this by using the -tt option and checking indent.log!

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 \{ OR $;
- then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the UnNamedGroupingBracesBrackets: follow field of the fine tuning section in Listing 556 on page 141.
5.9 Commands and the strings between their arguments

An example is shown in Listing 261 with default output given in Listing 262.

Example 69

Referencing Listing 262, there are three sets of unnamed braces. Note also that the maximum value of indentation is three tabs, and these come from:

- the \psforeach command’s mandatory argument;
- the first un-named braces mandatory argument;
- the first un-named braces body, which we define as any lines following the first opening \ or [ that defined the code block. This is the part controlled by the body field for noAdditionalIndent and friends from page 53.

Users wishing to customise the mandatory and/or optional arguments on a per-name basis for the UnNamedGroupingBracesBrackets should use always-un-named.

5.8.7.4 filecontents

code blocks behave just as environments, except that neither arguments nor items are sought.

5.8.8 Summary

Having considered all of the different types of code blocks, the functions of the fields given in Listings 263 and 264 should now make sense.

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 265.

### Listing 265: commandCodeBlocks

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>370</td>
<td><code>commandCodeBlocks:</code></td>
</tr>
<tr>
<td>371</td>
<td><code>roundParenthesesAllowed: 1</code></td>
</tr>
<tr>
<td>372</td>
<td><code>stringsAllowedBetweenArguments:</code></td>
</tr>
<tr>
<td>373</td>
<td><code>-</code></td>
</tr>
<tr>
<td>374</td>
<td><code>amalgamate: 1</code></td>
</tr>
<tr>
<td>375</td>
<td><code>- node</code></td>
</tr>
<tr>
<td>376</td>
<td><code>- at</code></td>
</tr>
<tr>
<td>377</td>
<td><code>- to</code></td>
</tr>
<tr>
<td>378</td>
<td><code>- decoration</code></td>
</tr>
<tr>
<td>379</td>
<td><code>- \+</code></td>
</tr>
<tr>
<td>380</td>
<td><code>- \-</code></td>
</tr>
<tr>
<td>381</td>
<td><code>- \/^\d</code></td>
</tr>
<tr>
<td>382</td>
<td><code>commandNameSpecial:</code></td>
</tr>
<tr>
<td>383</td>
<td><code>-</code></td>
</tr>
<tr>
<td>384</td>
<td><code>amalgamate: 1</code></td>
</tr>
<tr>
<td>385</td>
<td><code>- </code>@ifnextchar`{`</td>
</tr>
</tbody>
</table>

#### roundParenthesesAllowed: 0|1

**example 70**  

The need for this field was mostly motivated by commands found in code used to generate images in PSTricks and tikz; for example, let’s consider the code given in Listing 266.

### Listing 266: pstricks1.tex

```latex
\defFunction[algebraic]{torus}(u,v)
{(2+cos(u))*cos(v+\Pi)}
{(2+cos(u))*sin(v+\Pi)}
{sin(u)}
```

### Listing 267: pstricks1 default output

```latex
\defFunction[algebraic]{torus}(u,v)
{(2+cos(u))*cos(v+\Pi)}
{(2+cos(u))*sin(v+\Pi)}
{sin(u)}
```

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

By default, because `roundParenthesesAllowed` is set to 1 in Listing 265, then `latexindent.pl` will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 266, `latexindent.pl` finds all the arguments of `defFunction`, both before and after \((u,v)\).

The default output from running `latexindent.pl` on Listing 266 actually leaves it unchanged (see Listing 267); note in particular, this is because of `noAdditionalIndentGlobal` as discussed on page 64.

Upon using the YAML settings in Listing 269, and running the command

```
cmh:~$: latexindent.pl pstricks1.tex -l noRoundParentheses.yaml
```

we obtain the output given in Listing 268.
5.9 Commands and the strings between their arguments

Notice the difference between Listing 267 and Listing 268; in particular, in Listing 268, because round parentheses are not allowed, latexindent.pl finds that the \defFunction command finishes at the first opening round parenthesis. As such, the remaining braced, mandatory, arguments are found to be UnNamedGroupingBracesBrackets (see Table 2 on page 54) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 268.

example 71 Let’s explore this using the YAML given in Listing 271 and run the command

```
> latexindent.pl pstricks1.tex -l defFunction.yaml
```

then the output is as in Listing 270.

Notice in Listing 270 that the body of the defFunction command i.e, the subsequent lines containing arguments after the command name, have received the single space of indentation specified by Listing 271.

```
stringsAllowedBetweenArguments: (fields)
```

example 72 tikz users may well specify code such as that given in Listing 272; processing this code using latexindent.pl gives the default output in Listing 273.

With reference to Listing 265 on the preceding 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 272, it consumes:

- the optional argument [thin]
- the round-bracketed argument (c) because roundParenthesesAllowed is 1 by default
- the string to (specified in stringsAllowedBetweenArguments)
- the optional argument [in=110,out=-90]
- the string ++ (specified in stringsAllowedBetweenArguments)
5.9 Commands and the strings between their arguments

- the round-bracketed argument (0,-0.5cm) because roundParenthesesAllowed is 1 by default
- the string node (specified in stringsAllowedBetweenArguments)
- the optional argument [below,align=left,scale=0.5]

**example 73** We can explore this further, for example using Listing 275 and running the command

cmh:~$ latexindent.pl tikz-node1.tex -l draw.yaml

we receive the output given in Listing 274.

**Listing 274:** tikz-node1.tex using Listing 275

\draw[thin]
(c) to[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 274 has been given the appropriate two-spaces worth of indentation specified in Listing 275.

Let's compare this with the output from using the YAML settings in Listing 277, and running the command

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

given in Listing 276.

**Listing 276:** tikz-node1.tex using Listing 277

\draw[thin]
(c) to[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 54) 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 265 on page 72, 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 278 or Listing 279 is equivalent to using the settings in Listing 280.
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 281 means that only the strings specified in that field will be used.

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 279 to 281.

We may explore this feature further with the code in Listing 282, whose default output is given in Listing 283.

Let's compare this with the output from using the YAML settings in Listing 285, and running the command

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

given in Listing 284.

You might like to compare the output given in Listing 283 and Listing 284. Note, in particular, in
Listing 283 that the foreach command has not included any of the subsequent strings, and that the braces have been treated as a namedGroupingBracesBrackets. In Listing 284 the foreach command has been allowed to have \x/\y and in between arguments because of the settings given in Listing 285.

commandNameSpecial: {fields}

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 ].

element 75

For example, consider the sample file in Listing 286, which has default output in Listing 287.

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

Notice that in Listing 287 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 124.

For demonstration, we can compare this output with that given in Listing 288 in which the settings from Listing 289 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 amalgamate field can be used for commandNameSpecial, just as for stringsAllowedBetweenArguments. The same condition holds as stated previously, which we state again here:

Warning!

It is important to note that the amalgamate field, if used, in either commandNameSpecial or stringsAllowedBetweenArguments must be in the first field, and specified using the syntax given in Listings 279 to 281.
SECTION 6

The -m (modifylinebreaks) switch

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

6.1 Text Wrapping .................................................. 79
   6.1.1 Text wrap: overview ..................................... 79
   6.1.2 Text wrap: simple examples .............................. 80
   6.1.3 Text wrap: blocksFollow examples .................... 81
   6.1.4 Text wrap: blocksBeginWith examples .................. 85
   6.1.5 Text wrap: blocksEndBefore examples .................. 87
   6.1.6 Text wrap: trailing comments and spaces ............. 88
   6.1.7 Text wrap: when before/after ......................... 89
   6.1.8 Text wrap: wrapping comments ........................ 91
   6.1.9 Text wrap: huge, tabstop and separator .............. 92

6.2 oneSentencePerLine: modifying line breaks for sentences .......... 93
   6.2.1 oneSentencePerLine: overview .......................... 94
   6.2.2 oneSentencePerLine: sentencesFollow .................. 96
   6.2.3 oneSentencePerLine: sentencesBeginWith ............... 97
   6.2.4 oneSentencePerLine: sentencesEndWith ................ 98
   6.2.5 Features of the oneSentencePerLine routine .......... 99
   6.2.6 oneSentencePerLine: text wrapping and indenting sentences .... 101
   6.2.7 oneSentencePerLine: text wrapping and indenting sentences, when before/after 104
   6.2.8 oneSentencePerLine: text wrapping sentences and comments ...... 105

6.3 Poly-switches .................................................. 105
   6.3.1 Poly-switches for environments ......................... 106
      6.3.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-Line .... 106
      6.3.1.2 Adding line breaks: EndStartsOnOwnLine and EndFinishesWithLine-Break .......... 108
      6.3.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary .... 110
      6.3.1.4 Removing line breaks (poly-switches set to −1) ........ 111
      6.3.1.5 About trailing horizontal space ................... 113
      6.3.1.6 poly-switch line break removal and blank lines ............... 113
   6.3.2 Poly-switches for double backslash ..................... 115
      6.3.2.1 Double backslash starts on own line ................. 115
      6.3.2.2 Double backslash finishes with line break ............ 116
As of Version 3.0, latexindent.pl has the \texttt{-m} switch, which permits latexindent.pl to modify line breaks, according to the specifications in the \texttt{modifyLineBreaks} field. The settings in this field will only be considered if the \texttt{-m} switch has been used. A snippet of the default settings of this field is shown in Listing 290.

\begin{verbatim}
500 modifyLineBreaks: 501   preserveBlankLines: 1 # 0/1
502   condenseMultipleBlankLinesInto: 1 # 0/1
\end{verbatim}

Having read the previous paragraph, it should sound reasonable that, if you call latexindent.pl using the \texttt{-m} switch, then you give it permission to modify line breaks in your file, but let’s be clear:

\begin{center}
\textbf{Warning!}
\end{center}

If you call latexindent.pl with the \texttt{-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.

\begin{center}
\textbf{preserveBlankLines: 0|1}
\end{center}

This field is directly related to poly-switches, discussed in Section 6.3. 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 \texttt{condenseMultipleBlankLinesInto} is greater than 0, discussed next.

\begin{center}
\textbf{condenseMultipleBlankLinesInto: (positive integer)}
\end{center}

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.

\begin{center}
\textbf{example 76}
\end{center}

As an example, Listing 291 shows a sample file with blank lines; upon running

\begin{verbatim}
$ latexindent.pl myfile.tex \texttt{-m} \texttt{-o=+-mod1}
\end{verbatim}

the output is shown in Listing 292; note that the multiple blank lines have been condensed into one blank line, and note also that we have used the \texttt{-m} switch!
6.1 Text Wrapping

The text wrapping routine has been over-hauled as of V3.16; I hope that the interface is simpler, and most importantly, the results are better.

The complete settings for this feature are given in Listing 293.

6.1.1 Text wrap: overview

An overview of how the text wrapping feature works:

1. the default value of columns is 0, which means that text wrapping will not happen by default;
2. it happens after verbatim blocks have been found;
3. it happens after the oneSentencePerLine routine (see Section 6.2);
4. it can happen before or after all of the other code blocks are found and does not operate on a per-code-block basis; when using before this means that, including indentation, you may receive a column width wider than that which you specify in columns, and in which case you probably wish to explore after in Section 6.1.7;
5. code blocks to be text wrapped will:
(a) follow the fields specified in `blocksFollow`
(b) `begin` with the fields specified in `blocksBeginWith`
(c) `end` before the fields specified in `blocksEndBefore`

6. setting `columns` to a value > 0 will text wrap blocks by first removing line breaks, and then wrapping according to the specified value of `columns`;
7. setting `columns` to −1 will only remove line breaks within the text wrap block;
8. by default, the text wrapping routine will remove line breaks within text blocks because `removeBlockLineBreaks` is set to 1; switch it to 0 if you wish to change this;
9. about trailing comments within text wrap blocks:
   (a) trailing comments that do not have leading space instruct the text wrap routine to connect the lines without space (see Listing 331);
   (b) multiple trailing comments will be connected at the end of the text wrap block (see Listing 335);
   (c) the number of spaces between the end of the text wrap block and the (possibly combined) trailing comments is determined by the spaces (if any) at the end of the text wrap block (see Listing 337);
10. trailing comments can receive text wrapping; examples are shown in Section 6.1.8 and Section 6.2.8.

We demonstrate this feature using a series of examples.

### 6.1.2 Text wrap: simple examples

**Example 77** Let's use the sample text given in Listing 294.

```latex
\begin{verbatim}
Here is a line of text that will be wrapped by latexindent.pl.
Here is a line of text that will be wrapped by latexindent.pl.
\end{verbatim}
```

We will change the value of `columns` in Listing 296 and then run the command

```bash
\$ latexindent.pl -m -l textwrap1.yaml textwrap1.tex
```

then we receive the output given in Listing 295.

```latex
\begin{verbatim}
Here is a line of text that will be wrapped by latexindent.pl.
Here is a line of text that will be wrapped by latexindent.pl.
\end{verbatim}
```

**Example 78** If we set `columns` to −1 then `latexindent.pl` remove line breaks within the text wrap block, and will not perform text wrapping. We can use this to undo text wrapping.

Starting from the file in Listing 295 and using the settings in Listing 297
6.1 Text Wrapping

Listing 297: textwrap1A.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: -1
```

and running

```
cmh:~$ latexindent.pl -m -l textwrap1A.yaml textwrap1-mod1.tex
```

gives the output in Listing 298.

**Listing 298: textwrap1-mod1A.tex**

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

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

---

**Example 79**

By default, the text wrapping routine will convert multiple spaces into single spaces. You can change this behaviour by flicking the switch `multipleSpacesToSingle` which we have done in Listing 300.

Using the settings in Listing 300 and running

```
cmh:~$ latexindent.pl -m -l textwrap1B.yaml textwrap1-mod1.tex
```

gives the output in Listing 299.

**Listing 299: textwrap1-mod1B.tex**

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

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

---

We note that in Listing 299 the multiple spaces have *not* been condensed into single spaces.

---

6.1.3 Text wrap: blocksFollow examples

We examine the `blocksFollow` field of Listing 293.

**Example 80**

Let's use the sample text given in Listing 301.

**Listing 301: tw-headings1.tex**

```latex
\section{my heading}\label{mylabel1}
text to be wrapped from the first section
\subsection{subheading}
text to be wrapped from the first section
```

We note that Listing 301 contains the heading commands `section` and `subsection`. Upon running the command
then we receive the output given in Listing 302.

**Listing 302: tw-headings1-mod1.tex**

```latex
\section{my heading} \label{mylabel1}
text to be wrapped
from the first section
\subsection{subheading}
text to be wrapped
from the first section
```

We reference Listing 293 on page 79 and also Listing 158 on page 51:

- in Listing 293 the headings field is set to 1, which instructs latexindent.pl to read the fields from Listing 158 on page 51, regardless of the value of indentAfterThisHeading or level;
- the default is to assume that the heading command can, optionally, be followed by a label command.

If you find scenarios in which the default value of headings does not work, then you can explore the other field.

We can turn off headings as in Listing 304 and then run

```bash
$ latexindent.pl -m -l textwrap1.yaml,bf-no-headings.yaml tw-headings1.tex
```

gives the output in Listing 303, in which text wrapping has been instructed *not to happen* following headings.

**Listing 303: tw-headings1-mod2.tex**

```latex
\section{my heading} \label{mylabel1}
text to be wrapped from the first section
\subsection{subheading}
text to be wrapped from the first section
```

**Listing 304: bf-no-headings.yaml**

```yaml
modifyLineBreaks:
textWrapOptions:
blocksFollow:
headings: 0
```

---

**Example 81**

Let's use the sample text given in Listing 305.

**Listing 305: tw-comments1.tex**

```latex
/% trailing comment
text to be wrapped following first comment
/% another comment
text to be wrapped following second comment
```

We note that Listing 305 contains trailing comments. Upon running the command

```bash
$ latexindent.pl -m -l textwrap1.yaml tw-comments1.tex
```
then we receive the output given in Listing 306.

```latex
\begin{verbatim}
\% trailing comment
\text{text to be wrapped following first comment}
\% another comment
\text{text to be wrapped following second comment}
\end{verbatim}
```

With reference to Listing 293 on page 79 the `commentOnPreviousLine` field is set to 1, which instructs `latexindent.pl` to find text wrap blocks after a comment on its own line.

We can turn off comments as in Listing 308 and then run

```bash
$ latexindent.pl -m -l textwrap1.yaml,bf-no-comments.yaml tw-comments1.tex
```

gives the output in Listing 307, in which text wrapping has been instructed not to happen following comments on their own line.

```latex
\begin{verbatim}
\% trailing comment
\text{text to be wrapped following first comment}
\% another comment
\text{text to be wrapped following second comment}
\end{verbatim}
```

Referencing Listing 293 on page 79 the `blocksFollow` fields `par`, `blankline`, `verbatim` and `filecontents` fields operate in analogous ways to those demonstrated in the above.

The other field of the `blocksFollow` can either be 0 (turned off) or set as a regular expression. The default value is set to `\\]|\item(?:\h|\[)` which can be translated to `backslash followed by a square bracket or backslash item followed by horizontal space or a square bracket`, or in other words, `end of display math` or an item command.

**example 82** Let’s use the sample text given in Listing 309.

```latex
\begin{verbatim}
\% trailing comment
text to be wrapped following first comment
\% another comment
text to be wrapped following second comment
\end{verbatim}
```

We note that Listing 309 contains display math. Upon running the command

```bash
$ latexindent.pl -m -l textwrap1.yaml tw-disp-math1.tex
```

then we receive the output given in Listing 310.
6.1 Text Wrapping

With reference to Listing 293 on page 79 the other field is set to \[], which instructs \texttt{latexindent.pl} to find text wrap blocks after the end of display math.

We can turn off this switch as in Listing 312 and then run

```
cmh:\$ latexindent.pl -m -l textwrap1.yaml,bf-no-disp-math.yaml tw-disp-math1.tex
```

gives the output in Listing 311, in which text wrapping has been instructed \textit{not to happen} following display math.

```
LISTING 311: tw-disp-math1-mod2.tex
```

text to be wrapped
before display math
\[ y = x \]
text to be wrapped
after display math

Naturally, you should feel encouraged to customise this as you see fit.

The \texttt{blocksFollow} field \textit{deliberately} does not default to allowing text wrapping to occur after \texttt{begin environment} statements. You are encouraged to customize the other field to accommodate the environments that you would like to text wrap individually, as in the next example.

**example 83**

Let's use the sample text given in Listing 313.

```
LISTING 313: tw-bf-myenv1.tex
```

text to be wrapped before \texttt{myenv} environment
\begin{myenv}
text to be wrapped within \texttt{myenv} environment
\end{myenv}
text to be wrapped after \texttt{myenv} environment

We note that Listing 313 contains \texttt{myenv} environment. Upon running the command

```
cmh:\$ latexindent.pl -m -l textwrap1.yaml tw-bf-myenv1.tex
```

then we receive the output given in Listing 314.
6.1 Text Wrapping

We note that we have not received much text wrapping. We can turn do better by employing Listing 316 and then run

\begin{verbatim}
$ latexindent.pl -m -l textwrap1.yaml,tw-bf-myenv.yaml tw-bf-myenv1.tex
\end{verbatim}

which gives the output in Listing 315, in which text wrapping has been implemented across the file.

6.1.4 Text wrap: blocksBeginWith examples

We examine the blocksBeginWith field of Listing 293 with a series of examples.

example 84  By default, text wrap blocks can begin with the characters a–z and A–Z.

If we start with the file given in Listing 317

\begin{verbatim}
123 text to be wrapped before display math
\[
y = x\]
456 text to be wrapped after display math
\end{verbatim}

and run the command

\begin{verbatim}
$ latexindent.pl -m -l textwrap1.yaml tw-0-9.tex
\end{verbatim}

then we receive the output given in Listing 318 in which text wrapping has not occurred.
6.1 Text Wrapping

<table>
<thead>
<tr>
<th>LISTING 318: tw-0-9-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 text to be wrapped before display math [ y = x ]</td>
</tr>
<tr>
<td>456 text to be wrapped after display math</td>
</tr>
</tbody>
</table>

We can allow paragraphs to begin with 0-9 characters by using the settings in Listing 320 and running

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml,bb-0-9-yaml tw-0-9.tex
```
gives the output in Listing 319, in which text wrapping has happened.

<table>
<thead>
<tr>
<th>LISTING 319: tw-0-9-mod2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>123 text to be wrapped before display math [ y = x ]</td>
</tr>
<tr>
<td>456 text to be wrapped after display math</td>
</tr>
</tbody>
</table>

### Example 85

Let's now use the file given in Listing 321

<table>
<thead>
<tr>
<th>LISTING 321: tw-bb-announce1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>% trailing comment \announce{announce text} and text to be wrapped before goes here</td>
</tr>
</tbody>
</table>

and run the command

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml tw-bb-announce1.tex
```
then we receive the output given in Listing 322 in which text wrapping has not occurred.

<table>
<thead>
<tr>
<th>LISTING 322: tw-bb-announce1-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>% trailing comment \announce{announce text} and text to be wrapped before goes here</td>
</tr>
</tbody>
</table>

We can allow \announce to be at the beginning of paragraphs by using the settings in Listing 324 and running

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml,tw-bb-announce.yaml tw-bb-announce1.tex
```
gives the output in Listing 323, in which text wrapping has happened.
6.1.5 Text wrap: blocksEndBefore examples

We examine the blocksEndBefore field of Listing 293 with a series of examples.

**example 86**

Let's use the sample text given in Listing 325.

```
before equation
text
\begin{align}
  1 &amp; 2 \\
  3 &amp; 4
\end{align}
after equation
text
```

We note that Listing 325 contains an environment. Upon running the command

```
\texttt{cmh:~\$ latexindent.pl -m -l textwrap1A.yaml tw-be-equation.tex}
```

then we receive the output given in Listing 326.

```
before equation text
\begin{align}
  1 &amp; 2 \\
  3 &amp; 4
\end{align}
after equation text
```

With reference to Listing 293 on page 79 the other field is set to \begin{}\end{\}, which
instructs latexindent.pl to stop text wrap blocks before begin statements, display math, and end statements.

We can turn off this switch as in Listing 327 and then run

```
\texttt{cmh:~\$ latexindent.pl -m -l textwrap1A.yaml,tw-be-equation.yaml tw-be-equation.tex}
```

gives the output in Listing 328, in which text wrapping has been instructed *not* to stop at these statements.
6.1 Text Wrapping

6.1.6 Text wrap: trailing comments and spaces

We explore the behaviour of the text wrap routine in relation to trailing comments using the following examples.

example 87 The file in Listing 329 contains a trailing comment which does have a space in front of it.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc1.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output given in Listing 330.

```
LISTING 329: tw-tc1.tex
foo %
bar
```

```
LISTING 330: tw-tc1-mod1.tex
foo bar%
```

example 88 The file in Listing 331 contains a trailing comment which does not have a space in front of it.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc2.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 332.

```
LISTING 331: tw-tc2.tex
foo%
bar
```

```
LISTING 332: tw-tc2-mod1.tex
foobar%
```

We note that, because there is not a space before the trailing comment, that the lines have been joined without a space.

dexample 89 The file in Listing 333 contains multiple trailing comments.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc3.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 334.

```
LISTING 333: tw-tc3.tex
foo  %1
bar%2
three
```

```
LISTING 334: tw-tc3-mod1.tex
foo barthree%1%2
```

Naturally, you should feel encouraged to customise this as you see fit.
example 90  The file in Listing 335 contains multiple trailing comments.
Running the command

```
cmh:~$ latexindent.pl -m tw-tc4.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 336.

```
<table>
<thead>
<tr>
<th>LISTING 335: tw-tc4.tex</th>
<th>LISTING 336: tw-tc4-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo %1</td>
<td>foo barthree%1%2%3</td>
</tr>
<tr>
<td>bar%2</td>
<td></td>
</tr>
<tr>
<td>three%3</td>
<td></td>
</tr>
</tbody>
</table>
```

example 91  The file in Listing 337 contains multiple trailing comments.
Running the command

```
cmh:~$ latexindent.pl -m tw-tc5.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 338.

```
<table>
<thead>
<tr>
<th>LISTING 337: tw-tc5.tex</th>
<th>LISTING 338: tw-tc5-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo%1</td>
<td>foobarthree␣%1%2%3</td>
</tr>
<tr>
<td>bar%2</td>
<td></td>
</tr>
<tr>
<td>three␣%3</td>
<td></td>
</tr>
</tbody>
</table>
```
The space at the end of the text block has been preserved.

example 92  The file in Listing 339 contains multiple trailing comments.
Running the command

```
cmh:~$ latexindent.pl -m tw-tc6.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 340.

```
<table>
<thead>
<tr>
<th>LISTING 339: tw-tc6.tex</th>
<th>LISTING 340: tw-tc6-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo%1</td>
<td>foobar␣%1</td>
</tr>
<tr>
<td>bar</td>
<td></td>
</tr>
</tbody>
</table>
```
The space at the end of the text block has been preserved.

6.1.7  Text wrap: when before/after

The text wrapping routine operates, by default, before the code blocks have been found, but this can be changed to after:

- **before** means it is likely that the columns of wrapped text may exceed the value specified in columns;
- **after** means it columns of wrapped text should not exceed the value specified in columns.

We demonstrate this in the following examples. See also Section 6.2.7.

example 93  Let’s begin with the file in Listing 341.
This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
\begin{myenv}
This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
\end{myenv}

Using the settings given in Listing 343 and running the command
```
cmh:~$ latexindent.pl textwrap8.tex -o=+-mod1.tex -l=tw-before1.yaml -m
```
gives the output given in Listing 342.

We note that, in Listing 342, that the wrapped text has exceeded the specified value of columns (35) given in Listing 343. We can affect this by changing when; we explore this next.

**example 94** We continue working with Listing 341.

Using the settings given in Listing 345 and running the command
```
cmh:~$ latexindent.pl textwrap8.tex -o=+-mod2.tex -l=tw-after1.yaml -m
```
gives the output given in Listing 344.
6.1 Text Wrapping

This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
\begin{myenv}
This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
\end{myenv}

----|----|----|----|----|----|----|----|
5 10 15 20 25 30 35 40

We note that, in Listing 344, that the wrapped text has obeyed the specified value of columns (35) given in Listing 345.

6.1.8 Text wrap: wrapping comments

You can instruct latexindent.pl to apply text wrapping to comments; we demonstrate this with examples, see also Section 6.2.8.

example 95 We use the file in Listing 346 which contains a trailing comment block.

My first sentence
% first comment
% second
% third comment
% fourth

Using the settings given in Listing 348 and running the command

cmh:~$ latexindent.pl textwrap9.tex -o=+-mod1.tex -l=wrap-comments1.yaml -m

gives the output given in Listing 347.

We note that, in Listing 347, that the comments have been combined and wrapped because of the annotated line specified in Listing 348.

example 96 We use the file in Listing 349 which contains a trailing comment block.

My first sentence
% first comment second third
% comment fourth

Using the settings given in Listing 351 and running the command

...


6.1 Text Wrapping

---

**Listing 350:** textwrap10-mod1.tex

My first sentence
% first comment second third
% comment fourth

---|-----|-----|-----|-----|-----|-----|-----|
5 10 15 20 25 30 35 40

---

**Listing 351:** wrap-comments1.yaml

modifyLineBreaks:
textWrapOptions:
columns: 35
comments:
wrap: 1 #<!

---

We note that, in Listing 350, that the comments have been combined and wrapped because of the annotated line specified in Listing 351, and that the space from the leading comment has not been inherited; we will explore this further in the next example.

**Example 97** We continue to use the file in Listing 349.

Using the settings given in Listing 353 and running the command

```
cmh:~$ latexindent.pl textwrap10.tex -o=+-mod2.tex -l=wrap-comments2.yaml -m
```

gives the output given in Listing 352.

---

**Listing 352:** textwrap10-mod2.tex

My first sentence
% first comment second third
% comment fourth

---|-----|-----|-----|-----|-----|-----|-----|
5 10 15 20 25 30 35 40

---

**Listing 353:** wrap-comments2.yaml

modifyLineBreaks:
textWrapOptions:
columns: 35
comments:
wrap: 1 #<!
inheritLeadingSpace: 1 #<!

---

We note that, in Listing 352, that the comments have been combined and wrapped and that the leading space has been inherited because of the annotated lines specified in Listing 353.

**6.1.9 Text wrap: huge, tabstop and separator**

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 [46]. 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.

---

**Example 98** For example, using the settings in Listings 355 and 357 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 354 and 356.
6.2 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 [7] for helping to shape and test this feature. The behaviour of this part of the script is controlled by the switches detailed in Listing 361, all of which we discuss next.

You can also specify the `tabstop` field as an integer value, which is passed to the text wrap module; see [46] for details.

example 99

Starting with the code in Listing 358 with settings in Listing 359, and running the command

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

gives the code given in Listing 360.

You can specify `separator`, `break` and `unexpand` options in your settings in analogous ways to those demonstrated in Listings 357 and 359, and they will be passed to the `Text::Wrap` module. I have not found a useful reason to do this; see [46] for more details.
6.2 oneSentencePerLine: modifying line breaks for sentences

<table>
<thead>
<tr>
<th>Listing 361: oneSentencePerLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>503 oneSentencePerLine:</td>
</tr>
<tr>
<td>504   manipulateSentences: 0   # 0/1</td>
</tr>
<tr>
<td>505   removeSentenceLineBreaks: 1 # 0/1</td>
</tr>
<tr>
<td>506   multipleSpacesToSingle: 1 # 0/1</td>
</tr>
<tr>
<td>507   textWrapSentence: 0       # 1 disables main textWrap</td>
</tr>
<tr>
<td>508   sentenceIndent: &quot;&quot;</td>
</tr>
<tr>
<td>509   sentencesFollow:</td>
</tr>
<tr>
<td>510     par: 1                 # 0/1</td>
</tr>
<tr>
<td>511     blankLine: 1           # 0/1</td>
</tr>
<tr>
<td>512     fullStop: 1            # 0/1</td>
</tr>
<tr>
<td>513     exclamationMark: 1     # 0/1</td>
</tr>
<tr>
<td>514     questionMark: 1        # 0/1</td>
</tr>
<tr>
<td>515     rightBrace: 1         # 0/1</td>
</tr>
<tr>
<td>516     commentOnPreviousLine: 1 # 0/1</td>
</tr>
<tr>
<td>517     other: 0               # regex</td>
</tr>
<tr>
<td>518   sentencesBeginWith:</td>
</tr>
<tr>
<td>519     A-Z: 1                # 0/1</td>
</tr>
<tr>
<td>520     a-z: 0                # 0/1</td>
</tr>
<tr>
<td>521     other: 0              # regex</td>
</tr>
<tr>
<td>522   sentencesEndWith:</td>
</tr>
<tr>
<td>523     basicFullStop: 0       # 0/1</td>
</tr>
<tr>
<td>524     betterFullStop: 1      # 0/1</td>
</tr>
<tr>
<td>525     exclamationMark: 1     # 0/1</td>
</tr>
<tr>
<td>526     questionMark: 1       # 0/1</td>
</tr>
<tr>
<td>527     other: 0              # regex</td>
</tr>
</tbody>
</table>

### 6.2.1 oneSentencePerLine: overview

An overview of how the oneSentencePerLine routine feature works:

1. the default value of manipulateSentences is 0, which means that oneSentencePerLine will **not** happen by default;
2. it happens **after** verbatim blocks have been found;
3. it happens **before** the text wrapping routine (see Section 6.1);
4. it happens **before** the main code blocks have been found;
5. sentences to be found:
   - (a) **follow** the fields specified in sentencesFollow
   - (b) **begin** with the fields specified in sentencesBeginWith
   - (c) **end** with the fields specified in sentencesEndWith
6. by default, the oneSentencePerLine routine will remove line breaks within sentences because removeBlockLineBreaks is set to 1; switch it to 0 if you wish to change this;
7. sentences can be text wrapped according to textWrapSentence, and will be done either **before** or **after** the main indentation routine (see Section 6.2.7);
8. about trailing comments within text wrap blocks:
   - (a) multiple trailing comments will be connected at the end of the sentence;
   - (b) the number of spaces between the end of the sentence and the (possibly combined) trailing comments is determined by the spaces (if any) at the end of the sentence.

We demonstrate this feature using a series of examples.

**manipulateSentences: 0|1**

This is a binary switch that details if latexindent.pl should perform the sentence manipulation
6.2 oneSentencePerLine: modifying line breaks for sentences

The one-sentence-per-line 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.

\begin{verbatim}
removeSentenceLineBreaks: 0|1
\end{verbatim}

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

\textbf{example 100} For example, consider \texttt{multiple-sentences.tex} shown in Listing 362.

\begin{verbatim}
Listing 362: 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.
\end{verbatim}

If we use the YAML files in Listings 364 and 366, and run the commands

\begin{verbatim}
cmh:~$ latexindent.pl multiple-sentences -m -l=manipulate-sentences.yaml
\end{verbatim}

then we obtain the respective output given in Listings 363 and 365.

\begin{verbatim}
Listing 363: multiple-sentences.tex using Listing 364
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.
\end{verbatim}

\begin{verbatim}
Listing 364: manipulate-sentences.yaml
modifyLineBreaks:
oneSentencePerLine:
manipulateSentences: 1
\end{verbatim}

\begin{verbatim}
Listing 365: multiple-sentences.tex using Listing 366
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.
\end{verbatim}

\begin{verbatim}
Listing 366: keep-sen-line-breaks.yaml
modifyLineBreaks:
oneSentencePerLine:
manipulateSentences: 1
removeSentenceLineBreaks: 0
\end{verbatim}

Notice, in particular, that the 'internal' sentence line breaks in Listing 362 have been removed in Listing 363, but have not been removed in Listing 365.

\begin{verbatim}
multipleSpacesToSingle: 0|1
\end{verbatim}

By default, the one-sentence-per-line routine will convert multiple spaces into single spaces. You can change this behaviour by changing the switch \texttt{multipleSpacesToSingle} to a value of 0.
The remainder of the settings displayed in Listing 361 on page 94 instruct \texttt{latexindent.pl} on how to define a sentence. From the perspective of \texttt{latexindent.pl} a sentence must:

- follow a certain character or set of characters (see Listing 367); by default, this is either \texttt{\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 368); by default, this is only capital letters;
- end with a character (see Listing 369); by default, these are full stop/period (.), exclamation mark (!) and question mark (?)).

In each case, you can specify the \texttt{other} field to include any pattern that you would like; you can specify anything in this field using the language of regular expressions.

\begin{Verbatim}
\textbf{Listing 367: sentencesFollow}
\begin{verbatim}
509 sentencesFollow:
510   par:  1 # 0/1
511   blankLine: 1 # 0/1
512   fullStop: 1 # 0/1
513   exclamationMark: 1 # 0/1
514   questionMark: 1 # 0/1
515   rightBrace: 1 # 0/1
516   commentOnPreviousLine: 1 # 0/1
517   other: 0 # regex
\end{verbatim}
\end{Verbatim}

\begin{Verbatim}
\textbf{Listing 368: sentencesBeginWith}
\begin{verbatim}
518 sentencesBeginWith:
519   A-Z: 1 # 0/1
520   a-z: 0 # 0/1
521   other: 0 # regex
\end{verbatim}
\end{Verbatim}

\begin{Verbatim}
\textbf{Listing 369: sentencesEndWith}
\begin{verbatim}
522 sentencesEndWith:
523   basicFullStop: 0 # 0/1
524   betterFullStop: 1 # 0/1
525   exclamationMark: 1 # 0/1
526   questionMark: 1 # 0/1
527   other: 0 # regex
\end{verbatim}
\end{Verbatim}

\subsection{6.2.2 oneSentencePerLine: sentencesFollow}

Let’s explore a few of the switches in \texttt{sentencesFollow}.

\begin{example}

We start with Listing 362 on the previous page, and use the YAML settings given in Listing 371. Using the command

\begin{verbatim}
\texttt{cmh:~}\$ latexindent.pl multiple-sentences -m -l=sentences-follow1.yaml
\end{verbatim}

we obtain the output given in Listing 370.

\begin{Verbatim}
\textbf{Listing 370: multiple-sentences.tex}
\begin{verbatim}
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.
\end{verbatim}
\end{Verbatim}

Notice that, because \texttt{blankLine} is set to 0, \texttt{latexindent.pl} will not seek sentences following a blank line, and so the fourth sentence has not been accounted for.
6.2 oneSentencePerLine: modifying line breaks for sentences

We can explore the other field in Listing 367 with the .tex file detailed in Listing 372.

<table>
<thead>
<tr>
<th>Listing 372: 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
```

then we obtain the respective output given in Listings 373 and 374.

<table>
<thead>
<tr>
<th>Listing 373: multiple-sentences1.tex using Listing 364 on page 95</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 374: multiple-sentences1.tex using Listing 375</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 373 the first sentence after the ) has not been accounted for, but that following the inclusion of Listing 375, the output given in Listing 374 demonstrates that the sentence has been accounted for correctly.

6.2.3 oneSentencePerLine: 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 368), and we can use the other field to define sentences to begin with other characters.

We use the file in Listing 376.

<table>
<thead>
<tr>
<th>Listing 376: multiple-sentences2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the first sentence.</td>
</tr>
<tr>
<td>$a$ can represent a number. 7 is at the beginning of this sentence.</td>
</tr>
</tbody>
</table>

Upon running the following commands

```
cmh:~$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml
```

then we obtain the respective output given in Listings 377 and 378.
One Sentence Per Line: Modifying Line Breaks for Sentences

This is the first sentence.

\(a\) can represent a number. 7 is at the beginning of this sentence.

Notice that in Listing 377, the first sentence has been accounted for but that the subsequent sentences have not. In Listing 378, all of the sentences have been accounted for, because the other field in Listing 379 has defined sentences to begin with either \(\$\) or any numeric digit, 0 to 9.

6.2.4 One Sentence Per Line: SentencesEndWith

Example 104

Let's return to Listing 362 on page 95; we have already seen the default way in which latexindent.pl will operate on the sentences in this file in Listing 363 on page 95. We can populate the other field with any character that we wish; for example, using the YAML specified in Listing 381 and the command

```
cmb:~$ latexindent.pl multiple-sentences -m -l=sentences-end1.yaml
cmb:~$ latexindent.pl multiple-sentences -m -l=sentences-end2.yaml
```

then we obtain the output in Listing 380.
There is a subtle difference between the output in Listings 380 and 382; in particular, in Listing 380 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 383, and the associated output in Listing 382 reflects this.

Referencing Listing 369 on page 96, 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.

**example 105**

Let’s consider the file shown in Listing 384.

```
Listing 384: url.tex
This sentence, \url{tex.stackexchange.com/} finishes here. Second sentence.
```

Upon running the following commands

```bash
cmh:~$ latexindent.pl url -m -l=manipulate-sentences.yaml
```

we obtain the output given in Listing 385.

```
Listing 385: url.tex using Listing 364 on page 95
This sentence, \url{tex.stackexchange.com/} finishes here.
Second sentence.
```

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 556 on page 141.

The basicFullStop routine should probably be avoided in most situations, as it does not accommodate the specifications above.

**example 106**

For example, using the following command

```bash
cmh:~$ latexindent.pl url -m -l=alt-full-stop1.yaml
```

and the YAML in Listing 387 gives the output in Listing 386.

```
Listing 386: url.tex using Listing 387
This sentence, \url{tex.stackexchange.com/} finishes here. Second sentence.
```

Notice that the full stop within the URL has not been accommodated correctly because of the non-default settings in Listing 387.

**6.2.5 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.
example 107 For example, if we begin with the .tex file in Listing 388, and run the command

```
$ latexindent .pl multiple-sentences3 -m -l=manipulate-sentences.yaml
```

then we obtain the output in Listing 389.

<table>
<thead>
<tr>
<th>LISTING 388: multiple-sentences3.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first sentence continues after the verbatim \begin{verbatim}</td>
</tr>
<tr>
<td>there are sentences within this. These</td>
</tr>
<tr>
<td>will not be operated</td>
</tr>
<tr>
<td>upon by latexindent.pl.</td>
</tr>
<tr>
<td>\end{verbatim} and finishes here. Second sentence % a commented full stop.</td>
</tr>
<tr>
<td>contains trailing comments, which are ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 389: multiple-sentences3.tex using Listing 364 on page 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first sentence continues after the verbatim \begin{verbatim}</td>
</tr>
<tr>
<td>there are sentences within this. These</td>
</tr>
<tr>
<td>will not be operated</td>
</tr>
<tr>
<td>upon by latexindent.pl.</td>
</tr>
<tr>
<td>\end{verbatim} and finishes here. Second sentence contains trailing comments, which are ignored.</td>
</tr>
<tr>
<td>% a commented full stop.</td>
</tr>
</tbody>
</table>

example 108 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 390 and run the commands

```
$ latexindent .pl multiple-sentences4 -m -l=manipulate-sentences.yaml
$ latexindent .pl multiple-sentences4 -m -l=keep-sen-line-breaks.yaml
```

then we obtain the output in Listings 391 and 392.

<table>
<thead>
<tr>
<th>LISTING 390: multiple-sentences4.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize}</td>
</tr>
<tr>
<td>\item continues</td>
</tr>
<tr>
<td>\end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 391: multiple-sentences4.tex using Listing 364 on page 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize} \item continues \end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 392: multiple-sentences4.tex using Listing 366 on page 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize} \item continues \end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>
example 109  Once you've read Section 6.3, you will know that you can accommodate the removal of internal sentence line breaks by using the YAML in Listing 394 and the command

\texttt{\textasciitilde latexindent.pl multiple-sentences4 -m -l=item-rules2.yaml}

the output of which is shown in Listing 393.

<table>
<thead>
<tr>
<th>Listing 393: multiple-sentences4.tex using Listing 394</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize} \item continues \end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 394: item-rules2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine:</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
<tr>
<td>items:</td>
</tr>
<tr>
<td>ItemStartsOnOwnLine: 1</td>
</tr>
<tr>
<td>environments:</td>
</tr>
<tr>
<td>BeginStartsOnOwnLine: 1</td>
</tr>
<tr>
<td>BodyStartsOnOwnLine: 1</td>
</tr>
<tr>
<td>EndStartsOnOwnLine: 1</td>
</tr>
<tr>
<td>EndFinishesWithLineBreak: 1</td>
</tr>
</tbody>
</table>

6.2.6 oneSentencePerLine: text wrapping and indenting sentences

The oneSentencePerLine can be instructed to perform text wrapping and indentation upon sentences.

example 110  Let's use the code in Listing 395.

<table>
<thead>
<tr>
<th>Listing 395: multiple-sentences5.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>A distincao entre conteudo \textit{real} e conteudo \textit{intencional} esta relacionada, ainda, a distincao entre o conceito husserliano de \textit{experiencia} e o uso popular desse termo. No sentido comum, o \textit{experimentado} e um complexo de eventos exteriores, e o \textit{experimentar} consiste em percepcoes (alem de julgamentos e outros atos) nas quais tais eventos aparecem como objetos, e objetos frequentemente to the end.</td>
</tr>
</tbody>
</table>

Referencing Listing 397, and running the following command

```
\texttt{\textasciitilde latexindent.pl multiple-sentences5 -m -l=sentence-wrap1.yaml}
```

we receive the output given in Listing 396.

<table>
<thead>
<tr>
<th>Listing 396: multiple-sentences5.tex using Listing 397</th>
</tr>
</thead>
<tbody>
<tr>
<td>A distincao entre conteudo \textit{real} e conteudo \textit{intencional} esta relacionada, ainda, a distincao entre o conceito husserliano de \textit{experiencia} e o uso popular desse termo. No sentido comum, o \textit{experimentado} e um complexo de eventos exteriores, e o \textit{experimentar} consiste em percepcoes (alem de julgamentos e outros atos) nas quais tais eventos aparecem como objetos, e objetos frequentemente to the end.</td>
</tr>
</tbody>
</table>

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.
example 111 The indentation of sentences requires that sentences are stored as code blocks. This means that you may need to tweak Listing 369 on page 96. Let’s explore this in relation to Listing 398.

<table>
<thead>
<tr>
<th>LISTING 398: multiple-sentences6.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider the following:</td>
</tr>
<tr>
<td>\begin{itemize}</td>
</tr>
<tr>
<td>\item firstly.</td>
</tr>
<tr>
<td>\item secondly.</td>
</tr>
<tr>
<td>\end{itemize}</td>
</tr>
</tbody>
</table>

By default, latexindent.pl will find the full-stop within the first item, which means that, upon running the following commands

```cmh```
$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
```
cmh::$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
```

```cmh```
y="modifyLineBreaks:oneSentencePerLine:sentenceIndent:''"
```

we receive the respective output in Listing 399 and Listing 400.

<table>
<thead>
<tr>
<th>LISTING 399: multiple-sentences6-mod1.tex using Listing 397</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider the following: \begin{itemize} \item firstly.</td>
</tr>
<tr>
<td>\item secondly.</td>
</tr>
<tr>
<td>\end{itemize}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 400: multiple-sentences6-mod2.tex using Listing 397 and no sentence indentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider the following: \begin{itemize} \item firstly.</td>
</tr>
<tr>
<td>\item secondly.</td>
</tr>
<tr>
<td>\end{itemize}</td>
</tr>
</tbody>
</table>

We note that Listing 399 the itemize code block has not been indented appropriately. This is because the oneSentencePerLine has been instructed to store sentences (because Listing 397); each sentence is then searched for code blocks.

example 112 We can tweak the settings in Listing 369 on page 96 to ensure that full stops are not followed by item commands, and that the end of sentences contains \end{itemize} as in Listing 401. This setting is actually an appended version of the betterFullStop from the fineTuning, detailed in Listing 556 on page 141.
6.2 oneSentencePerLine: modifying line breaks for sentences

Upon running

cmh:~$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml,itemize.yaml

we receive the output in Listing 402.

Consider the following: \begin{itemize}
\item firstly.
\item secondly.
\end{itemize}

Notice that the sentence has received indentation, and that the \texttt{itemize} code block has been found and indented correctly.

Text wrapping when using the oneSentencePerLine routine determines if it will remove line breaks while text wrapping, from the value of removeSentenceLineBreaks.
6.2.7 oneSentencePerLine: text wrapping and indenting sentences, when before/after

The text wrapping routine operates, by default, before the code blocks have been found, but this can be changed to after:

- before means it is likely that the columns of wrapped text may exceed the value specified in columns;
- after means it columns of wrapped text should not exceed the value specified in columns.

We demonstrate this in the following examples. See also Section 6.1.7.

example 113  Let's begin with the file in Listing 403.

<table>
<thead>
<tr>
<th>Listing 403: multiple-sentences8.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>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
</tbody>
</table>

Using the settings given in Listing 405 and running the command

```
$ latexindent -m ~/.latexindent/pl multiple-sentences8 -o=+-mod1.tex -l=sentence-wrap2 -m
```

gives the output given in Listing 404.

<table>
<thead>
<tr>
<th>Listing 404: multiple-sentences8-mod1.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>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
</tbody>
</table>

----|----|----|----|----|----|----|----|
 5 10 15 20 25 30 35 40

We note that, in Listing 404, that the wrapped text has exceeded the specified value of columns (35) given in Listing 405. We can affect this by changing when; we explore this next.

example 114  We continue working with Listing 403.

Using the settings given in Listing 407 and running the command

```
$ latexindent -m ~/.latexindent/pl multiple-sentences8.tex -o=+-mod2.tex -l=sentence-wrap3 -m
```

gives the output given in Listing 406.
6.3 Poly-switches

Every other field in the modifyLineBreaks field uses poly-switches, and can take one of the following integer values:

6.2.8 oneSentencePerLine: text wrapping sentences and comments

We demonstrate the one sentence per line routine with respect to text wrapping comments. See also Section 6.1.8.

example 115 Let's begin with the file in Listing 408.

Using the settings given in Listing 410 and running the command

```
cmh:~$ latexindent.pl multiple-sentences9 -o=+-mod1.tex -l=sentence-wrap4 -m
```
gives the output given in Listing 409.

We note that, in Listing 409, that the sentences have been wrapped, and so too have the comments because of the annotated line in Listing 410.

6.3 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 54. 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.

### 6.3.1 Poly-switches for environments

We start by viewing a snippet of defaultSettings.yaml in Listing 411; note that it contains global settings (immediately after the environments field) and that per-name settings are also allowed – in the case of Listing 411, settings for equation* have been specified for demonstration. Note that all poly-switches are off (set to 0) by default.

```
LISTING 411: environments

environments:
  BeginStartsOnOwnLine: 0  # -1,0,1,2,3,4
  BodyStartsOnOwnLine: 0  # -1,0,1,2,3,4
  EndStartsOnOwnLine: 0  # -1,0,1,2,3,4
  EndFinishesWithLineBreak: 0  # -1,0,1,2,3,4

equation*:
  BeginStartsOnOwnLine: 0  # -1,0,1,2,3,4
  BodyStartsOnOwnLine: 0  # -1,0,1,2,3,4
  EndStartsOnOwnLine: 0  # -1,0,1,2,3,4
  EndFinishesWithLineBreak: 0  # -1,0,1,2,3,4
```

Let’s begin with the simple example given in Listing 412; note that we have annotated key parts of the file using ♠, ♥, ♦ and ♣, these will be related to fields specified in Listing 411.

```
LISTING 412: env-mlb1.tex

before words♠ \begin{myenv}♥ body of myenv♦ \end{myenv}♣ after words
```

### 6.3.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine

**example 116** Let’s explore BeginStartsOnOwnLine and BodyStartsOnOwnLine in Listings 413 and 414, and in particular, let’s allow each of them in turn to take a value of 1.

```
LISTING 413: env-mlb1.yaml
modifyLineBreaks:
  environments:
    BeginStartsOnOwnLine: 1

LISTING 414: env-mlb2.yaml
modifyLineBreaks:
  environments:
    BodyStartsOnOwnLine: 1
```
After running the following commands,

```bash
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb1.yaml
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb2.yaml
```

the output is as in Listings 415 and 416 respectively.

There are a couple of points to note:

- in Listing 415 a line break has been added at the point denoted by ♠ in Listing 412; no other line breaks have been changed;
- in Listing 416 a line break has been added at the point denoted by ♥ in Listing 412; furthermore, note that the body of `myenv` has received the appropriate (default) indentation.

**example 117**

Let's now change each of the 1 values in Listings 413 and 414 so that they are 2 and save them into `env-mlb3.yaml` and `env-mlb4.yaml` respectively (see Listings 417 and 418).

Upon running the commands

```bash
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb3.yaml
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb4.yaml
```

we obtain Listings 419 and 420.

Note that line breaks have been added as in Listings 415 and 416, 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.

**example 118**

Let's now change each of the 1 values in Listings 413 and 414 so that they are 3 and save them into `env-mlb5.yaml` and `env-mlb6.yaml` respectively (see Listings 421 and 422).

Upon running the commands

```bash
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb5.yaml
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb6.yaml
```

we obtain Listings 423 and 424.
Note that line breaks have been added as in Listings 415 and 416, but this time a blank line has been added after adding the line break.

Let's now change each of the 1 values in Listings 421 and 422 so that they are 4 and save them into env-beg4.yaml and env-body4.yaml respectively (see Listings 425 and 426).

We will demonstrate this poly-switch value using the code in Listing 427.

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 428 and 429.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 428 a blank line has been inserted before the `\begin` statement, even though the `\begin` statement was already on its own line;
2. in Listing 429 a blank line has been inserted before the beginning of the `body`, even though it already began on its own line.

### 6.3.1.2 Adding line breaks: EndStartsOnOwnLine and EndFinishesWithLineBreak

Let's explore `EndStartsOnOwnLine` and `EndFinishesWithLineBreak` in Listings 430 and 431, and in particular, let's allow each of them in turn to take a value of 1.
After running the following commands,

\texttt{\$ latexindent.pl -m env-mlb.tex -l env-mlb7.yaml} \texttt{\$ latexindent.pl -m env-mlb.tex -l env-mlb8.yaml}

the output is as in Listings 432 and 433.

There are a couple of points to note:

- in Listing 432 a line break has been added at the point denoted by ♦ in Listing 412 on page 106; no other line breaks have been changed and the \texttt{\end{myenv}} statement has not received indentation (as intended);
- in Listing 433 a line break has been added at the point denoted by ♣ in Listing 412 on page 106.

\textbf{example 121} Let’s now change each of the 1 values in Listings 430 and 431 so that they are 2 and save them into env-mlb9.yaml and env-mlb10.yaml respectively (see Listings 434 and 435).

Upon running the commands

\texttt{\$ latexindent.pl -m env-mlb.tex -l env-mlb9.yaml} \texttt{\$ latexindent.pl -m env-mlb.tex -l env-mlb10.yaml}

we obtain Listings 436 and 437.

Note that line breaks have been added as in Listings 432 and 433, 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.

\textbf{example 122} Let’s now change each of the 1 values in Listings 430 and 431 so that they are 3 and save them into env-mlb11.yaml and env-mlb12.yaml respectively (see Listings 438 and 439).
Upon running the commands

```bash
cmh:$ latexindent.pl -m env-mlb11.tex -l env-mlb11.yaml
cmh:$ latexindent.pl -m env-mlb12.tex -l env-mlb12.yaml
```

we obtain Listings 440 and 441.

Note that line breaks have been added as in Listings 432 and 433, and that a blank line has been added after the line break.

**Example 123**

Let's now change each of the 1 values in Listings 438 and 439 so that they are 4 and save them into `env-end4.yaml` and `env-end-f4.yaml` respectively (see Listings 442 and 443).

We will demonstrate this poly-switch value using the code from Listing 427 on page 108.

Upon running the commands

```bash
cmh:$ latexindent.pl -m env-mlb11.tex -l env-end4.yaml
cmh:$ latexindent.pl -m env-mlb11.tex -l env-end-f4.yaml
```

then we receive the respective outputs in Listings 444 and 445.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 444 a blank line has been inserted before the `\end` statement, even though the `\end` statement was already on its own line;
2. in Listing 445 a blank line has been inserted after the `\end` statement, even though it already began on its own line.

### 6.3.1.3 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.
example 124  For example, if you process the file in Listing 446 using poly-switch values of 1, 2, or 3, it will be left unchanged.

\begin{myenv}
\text{body of myenv}
\end{myenv}

example 125  In contrast, the output from processing the file in Listing 447 will vary depending on the poly-switches used; in Listing 448 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 449 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 447 and by setting the other poly-switches considered so far to 2 in turn.

\begin{myenv} \%
\text{body of myenv}\%
\end{myenv} \%

The details of the discussion in this section have concerned global poly-switches in the environments field; each switch can also be specified on a per-name basis, which would take priority over the global values; with reference to Listing 411 on page 106, an example is shown for the equation* environment.

6.3.1.4 Removing line breaks (poly-switches set to \texttt{-1})

Setting poly-switches to \texttt{-1} tells latexindent.pl to remove line breaks of the \texttt{<part of the thing>}, if necessary.

example 126  We will consider the example code given in Listing 450, noting in particular the positions of the line break highlighters, ♠, ♥, ♦ and ♣, together with the associated YAML files in Listings 451 to 454.
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 455 to 458.

Notice that in:

- Listing 455 the line break denoted by ♠ in Listing 450 has been removed;
- Listing 456 the line break denoted by ♥ in Listing 450 has been removed;
- Listing 457 the line break denoted by ♦ in Listing 450 has been removed;
- Listing 458 the line break denoted by ♣ in Listing 450 has been removed.

We examined each of these cases separately for clarity of explanation, but you can combine all of the YAML settings in Listings 451 to 454 into one file; alternatively, you could tell `latexindent.pl` to load them all by using the following command, for example
6.3 Poly-switches

6.3.1.5 About trailing horizontal space

Recall that on page 33 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` is particularly relevant when considering the `-m` switch.

**example 127** We consider the file shown in Listing 459, which highlights trailing spaces.

```latex
\begin{myenv}
body of myenv
\end{myenv}
```

The output from the following commands

```bash
cmh:~$ latexindent.pl -m env-mlb5.tex -l env-mlb13,env-mlb14,env-mlb15,env-mlb16
```

is shown, respectively, in Listings 461 and 462; note that the trailing horizontal white space has been preserved (by default) in Listing 461, while in Listing 462, it has been removed using the switch specified in Listing 460.

**Listing 461:** `env-mlb5.tex` using Listings 455 to 458

```latex
\begin{myenv}
body of myenv\end{myenv}
```

**Listing 462:** `env-mlb5.tex` using Listings 455 to 458 and Listing 460

```latex
\begin{myenv}body of myenv\end{myenv}
```

6.3.1.6 poly-switch line break removal and blank lines

**example 128** Now let's consider the file in Listing 463, which contains blank lines.

```latex
\begin{myenv}
body of myenv\end{myenv}
```

Upon running the following commands

```bash
cmh:~$ latexindent.pl -m env-mlb6.tex -l env-mlb5,env-mlb13,env-mlb14,env-mlb15,env-mlb16,removeTWS-before
```

```
```
we receive the respective outputs in Listings 465 and 466. In Listing 465 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 466, we have allowed the poly-switches to remove blank lines because, in Listing 464, we have set preserveBlankLines to 0.

\textbf{example 129} We can explore this further using the blank-line poly-switch value of 3; let's use the file given in Listing 467.

Upon running the following commands

we receive the outputs given in Listings 468 and 469.

Notice that in:

- Listing 468 that \texttt{end\{one\}} has added a blank line, because of the value of EndFinishesWithLineBreak in Listing 439 on page 110, and even though the line break ahead of \texttt{begin\{two\}} should have been removed (because of BeginStartsOnOwnLine in Listing 451 on page 112), the blank line has been preserved by default;
- Listing 469, by contrast, has had the additional line-break removed, because of the settings in Listing 464.
6.3.2 Poly-switches for double backslash

With reference to lookForAlignDelims (see Listing 59 on page 33) you can specify poly-switches to dictate the line-break behaviour of double backslashes in environments (Listing 61 on page 34), commands (Listing 95 on page 40), or special code blocks (Listing 140 on page 47).

Consider the code given in Listing 470.

\begin{tabular}{cc}
1 & 2 \*\*\* \-\-\
3 & 4 \*\*\* \-\-\-
\end{tabular}

Referencing Listing 470:

- DBS stands for double backslash;
- line breaks ahead of the double backslash are annotated by \*, and are controlled by DBS\_Starts\_On\_Own\_Line;
- line breaks after the double backslash are annotated by \-, and are controlled by DBS\_Finishes\_With\_Line\_Break.

Let's explore each of these in turn.

### 6.3.2.1 Double backslash starts on own line

**example 130** We explore DBS\_Starts\_On\_Own\_Line (\* in Listing 470); starting with the code in Listing 470, together with the YAML files given in Listing 472 and Listing 474 and running the following commands

```
cmh:~\$ latexindent.pl -m tabular3.tex -l DBS1.yaml
cmh:~\$ latexindent.pl -m tabular3.tex -l DBS2.yaml
```

then we receive the respective output given in Listing 471 and Listing 473.

We note that

- Listing 472 specifies DBS\_Starts\_On\_Own\_Line for every environment (that is within lookForAlignDelims, Listing 62 on page 34); the double backslashes from Listing 470 have been moved to their own line in Listing 471;
- Listing 474 specifies DBS\_Starts\_On\_Own\_Line on a per-name basis for tabular (that is within lookForAlignDelims, Listing 62 on page 34); the double backslashes from Listing 470 have been moved to their own line in Listing 473, having added comment symbols before

---

6There is no longer any need for the code block to be specified within lookForAlignDelims for DBS poly-switches to activate.
moving them.

**example 131**  
We can combine DBS poly-switches with, for example, the `alignContentAfterDoubleBackSlash` in Section 5.5.6 on page 45.

For example, starting with the file Listing 475, and using the settings in Listings 131 and 133 on page 46 and running

```bash
cmh:~$ latexindent.pl -s -m -l alignContentAfterDBS1.yaml,DBS1.yaml tabular6.tex -o=+-mod1
cmh:~$ latexindent.pl -s -m -l alignContentAfterDBS2.yaml,DBS1.yaml tabular6.tex -o=+-mod2
```

gives the respective outputs shown in Listings 476 and 477.

<table>
<thead>
<tr>
<th>Listing 475: tabular6.tex</th>
<th>Listing 476: tabular6-mod1.tex</th>
<th>Listing 477: tabular6-mod2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc} 1 &amp; 2 2 \ 333 &amp; 4444 \ 55555 &amp; 666666 \end{tabular}</td>
<td>\begin{tabular}{cc} 1 &amp; 22 \ 333 &amp; 4444 \ 55555 &amp; 666666 \end{tabular}</td>
<td>\begin{tabular}{cc} 1 &amp; 22 \ 333 &amp; 4444 \ 55555 &amp; 666666 \end{tabular}</td>
</tr>
</tbody>
</table>

We note that:

- in Listing 476 the content after the double back slash has been aligned;
- in Listing 477 we see that 3 spaces have been added after the double back slash.

### 6.3.2.2 Double backslash finishes with line break

**example 132**  
Let’s now explore `DBSFinishesWithLineBreak` (□ in Listing 470); starting with the code in Listing 470, together with the YAML files given in Listing 479 and Listing 481 and running the following commands

```bash
cmh:~$ latexindent.pl -m tabular3.tex -l DBS3.yaml
cmh:~$ latexindent.pl -m tabular3.tex -l DBS4.yaml
```

then we receive the respective output given in Listing 478 and Listing 480.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc} 1 &amp; 2 \ 3 &amp; 4 \end{tabular}</td>
<td>modifyLineBreaks:</td>
<td>\begin{tabular}{cc} 1 &amp; 2 \ 3 &amp; 4 \end{tabular}</td>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td></td>
<td>environments:</td>
<td></td>
<td>environments:</td>
</tr>
<tr>
<td></td>
<td>DBSFinishesWithLineBreak: 1</td>
<td></td>
<td>DBSFinishesWithLineBreak: 2</td>
</tr>
</tbody>
</table>

We note that:

- Listing 479 specifies `DBSFinishesWithLineBreak` for every environment (that is within `lookForAlignDelims`, Listing 62 on page 34); the code following the double backslashes from Listing 470 has been moved to their own line in Listing 478;
6.3 Poly-switches

• Listing 481 specifies DBSFinishesWithLineBreak on a *per-name* basis for tabular (that is within lookForAlignDelims, Listing 62 on page 34); the first double backslashes from Listing 470 have moved code following them to their own line in Listing 480, having added comment symbols before moving them; the final double backslashes have *not* added a line break as they are at the end of the body within the code block.

### 6.3.2.3 Double backslash poly-switches for specialBeginEnd

**example 133**  Let’s explore the double backslash poly-switches for code blocks within specialBeginEnd code blocks (Listing 138 on page 47); we begin with the code within Listing 482.

**Listing 482: special4.tex**

```latex
\langle a & = b \\ & = c \\ & = d \\ & = e \rangle
```

Upon using the YAML settings in Listing 484, and running the command

```
$ latexindent -m special4.tex -l DBS5.yaml
```

then we receive the output given in Listing 483.

**Listing 483: special4.tex using Listing 484**

```
\langle 
  a & = b \\
  & = c \\
  & = d \\
  & = e \\
\rangle
```

**Listing 484: DBS5.yaml**

```
specialBeginEnd:
  cmhMath:
    lookForThis: 1
    begin: '\\<'
    end: '\\>'
  lookForAlignDelims:
    cmhMath: 1
  modifyLineBreaks:
    cmhMath: 1
    specialBeginEnd:
      DBSFinishesWithLineBreak: 1
      SpecialBodyStartsOnOwnLine: 1
      SpecialEndStartsOnOwnLine: 2
```

There are a few things to note:

• in Listing 484 we have specified cmhMath within lookForAlignDelims; without this, the double backslash poly-switches would be ignored for this code block;

• the DBSFinishesWithLineBreak poly-switch has controlled the line breaks following the double backslashes;

• 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.

### 6.3.2.4 Double backslash poly-switches for optional and mandatory arguments

For clarity, we provide a demonstration of controlling the double backslash poly-switches for optional and mandatory arguments.

**example 134**  We use with the code in Listing 485.
Upon using the YAML settings in Listings 487 and 489, 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 486 and 488.

6.3.2.5 Double backslash optional square brackets
The pattern matching for the double backslash will also, optionally, allow trailing square brackets that contain a measurement of vertical spacing, for example `\[3pt\].

example 135
For example, beginning with the code in Listing 490

```
\begin{pmatrix}
1 & 2 \[2pt] 3 & 4 \[3ex] 5 & 6 \[4pt] 7 & 8
\end{pmatrix}
```

and running the following command, using Listing 479,

```
$ latexindent.pl -m pmatrix3.tex -l DBS3.yaml
```

then we receive the output given in Listing 491.
You can customise the pattern for the double backslash by exploring the *fine tuning* field detailed in Listing 556 on page 141.

### 6.3.3 Poly-switches for other code blocks

Rather than repeat the examples shown for the environment code blocks (in Section 6.3.1 on page 106), 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 96 on page 41) can *not* be modified using `latexindent.pl`. However, there are two poly-switches available for `verbatim` code blocks: environments (Listing 38 on page 29), commands (Listing 39 on page 29) and `specialBeginEnd` (Listing 152 on page 50).
### 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♣ \begin{myenv}♥ body of myenv♦ \end{myenv}♣ after words</td>
<td>♣ BeginStartsOnOwnLine ♥ BodyStartsOnOwnLine ♦ EndStartsOnOwnLine ♣ EndFinishesWithLineBreak</td>
</tr>
<tr>
<td>ifelsefi</td>
<td>before words♣ \if...♥ body of if/or statement▲ \or▼ body of if/or statement★ \else□ body of else statement♦ \fi♣ after words</td>
<td>♣ IfStartsOnOwnLine ♥ BodyStartsOnOwnLine ▲ OrStartsOnOwnLine ▼ OrFinishesWithLineBreak ★ ElseStartsOnOwnLine □ ElseFinishesWithLineBreak ♣ FiStartsOnOwnLine ♣ FiFinishesWithLineBreak</td>
</tr>
<tr>
<td>optionalArguments</td>
<td>...♣ [♥ value before comma★, □ end of body of opt arg♦ ]♣ ...</td>
<td>♣ LSqBStartsOnOwnLine7 ♥ OptArgBodyStartsOnOwnLine ▲ CommaStartsOnOwnLine □ CommaFinishesWithLineBreak ♦ RSqBStartsOnOwnLine ♣ RSqBFinishesWithLineBreak</td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>...♣ {♥ value before comma★, □ end of body of mand arg♦ }♣ ...</td>
<td>♣ LCuBStartsOnOwnLine8 ♥ MandArgBodyStartsOnOwnLine ▲ CommaStartsOnOwnLine □ CommaFinishesWithLineBreak ♦ RCuBStartsOnOwnLine ♣ RCuBFinishesWithLineBreak</td>
</tr>
<tr>
<td>commands</td>
<td>before words♣ \mycommand♥ (arguments)</td>
<td>♣ CommandStartsOnOwnLine ♥ CommandNameFinishesWithLineBreak</td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>before words♣ myname♥ {braces/brackets}</td>
<td>♣ NameStartsOnOwnLine ♥ NameFinishesWithLineBreak</td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>before words♣ key♥ =♥ {braces/brackets}</td>
<td>♣ KeyStartsOnOwnLine ♥ EqualsStartsOnOwnLine ♥ EqualsFinishesWithLineBreak</td>
</tr>
<tr>
<td>items</td>
<td>before words♣ \item♥ ...</td>
<td>♣ ItemStartsOnOwnLine ♥ ItemFinishesWithLineBreak</td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>before words♣ {♥ body of special/middle★ \middle□ body of special/middle♦ }♣ after words</td>
<td>♣ SpecialBeginStartsOnOwnLine ♥ SpecialBodyStartsOnOwnLine □ SpecialMiddleStartsOnOwnLine □ SpecialMiddleFinishesWithLineBreak ♣ SpecialEndStartsOnOwnLine ♣ SpecialEndFinishesWithLineBreak</td>
</tr>
<tr>
<td>verbatim</td>
<td>before words♣ \begin{verbatim}</td>
<td>♣ VerbatimBeginStartsOnOwnLine</td>
</tr>
</tbody>
</table>

---

7 LSqB stands for Left Square Bracket  
8 LCuB stands for Left Curly Brace
6.3.4 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 previous page) and LCuBStartsOnownLine for mandatory arguments, and LSqBStartsOnOwnLine for optional arguments.

example 136 Let’s begin with the code in Listing 492 and the YAML settings in Listing 494; with reference to Table 3 on the preceding page, the key CommandNameFinishesWithLineBreak is an alias for BodyStartsOnOwnLine.

Upon running the command

```latex
\texttt{\textasciitilde latexindent.pl \textdollar m -l=mycom-mlb1.yaml mycommand1.tex}
```

we obtain Listing 493; note that the second mandatory argument beginning brace { has had its leading line break removed, but that the first brace has not.

```latex
\texttt{\textasciitilde latexindent.pl \textdollar m -l=mycom-mlb2.yaml mycommand1.tex}
```

we obtain Listing 495; both beginning braces { have had their leading line breaks removed.

example 137 Now let’s change the YAML file so that it is as in Listing 496; upon running the command

```latex
\texttt{\textasciitilde latexindent.pl \textdollar m -l=mycom-mlb2.yaml mycommand1.tex}
```

we obtain Listing 495; both beginning braces { have had their leading line breaks removed.

example 138 Now let’s change the YAML file so that it is as in Listing 498; upon running the command
6.3 Poly-switches

6.3.5 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches.

\begin{example}
We use the example from Listing 492 on the previous page, and consider the YAML settings given in Listing 500. The output from running
\begin{verbatim}
cmh:~$ latexindent.pl -m -l=mycom-mlb4.yaml mycommand1.tex
\end{verbatim}
is given in Listing 500.

\begin{verbatim}
\mycommand{mand arg text}{mand arg text}{mand arg text}
\end{verbatim}
\end{example}

Studying Listing 500, 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 499, 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.

\begin{example}
We can explore this further by considering the YAML settings in Listing 502; upon running the command
\begin{verbatim}
cmh:~$ latexindent.pl -m -l=mycom-mlb5.yaml mycommand1.tex
\end{verbatim}
we obtain the output given in Listing 501.

\begin{verbatim}
modifyLineBreaks:
  commands:
    CommandNameFinishesWithLineBreak: -1
  mandatoryArguments:
    LCuBStartsOnOwnLine: 1
\end{verbatim}
6.3 Poly-switches

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 504, and run the command

```bash
cmh:~$ latexindent.pl -m -l=mycom-mlb6.yaml mycommand1.tex
```

which gives the output in Listing 503.

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.

6.3.6 Conflicting poly-switches: nested code blocks

Example 141 Now let’s consider an example when nested code blocks have conflicting poly-switches; we’ll use the code in Listing 505, noting that it contains nested environments.

```latex
\begin{one}
one text
\begin{two}
two text
\end{two}
\end{one}
```

Let’s use the YAML settings given in Listing 507, which upon running the command

```bash
cmh:~$ latexindent.pl -m -l=nested-env-mlb1.yaml nested-env.tex
```

gives the output in Listing 506.
In Listing 506, 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 505, the \begin{two} environment is found before the \begin{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 506, this means that during Phase 1:

- the \begin{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 \begin{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 \begin{one} environment will be found and processed first, followed by the \begin{two} environment. If the \begin{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}).

**example 142**

We can explore this further using the poly-switches in Listing 509; upon running the command

```
cmh:~$ latexindent.pl -m -l=nested-env-mlb1.yaml nested-env-mlb2.yaml nested-env.tex
```

we obtain the output given in Listing 508.
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 -r switch will perform indentation and replacements, not respecting verbatim code blocks;
- the -rv switch will perform indentation and replacements, and will respect verbatim code blocks;
- the -rr switch will not perform indentation, and will perform replacements not respecting verbatim code blocks.

We will demonstrate each of the -r, -rv and -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>x</td>
</tr>
<tr>
<td>-rv</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>-rr</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

The default value of the replacements field is shown in Listing 510; 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 -r, -rv or -rr switches are active; when discussing YAML settings related to the replacement-mode switches, we will use the style given in Listing 510.

```yaml
replacements:
- amalgamate: 1
- this: latexindent.pl
  that: pl.latexindent
  lookForThis: 0
  when: before
```

The first entry within the replacements field is amalgamate, and is 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 one entry in the replacements field, but it can take as many entries as you would like; each one needs to begin with a - 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.

example 143

Beginning with the code in Listing 511 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 144 We begin with code given in Listing 515

```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.
7.3 Examples of replacements

Using the YAML in Listing 517, and running the command

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

then we achieve the output in Listing 516.

Note that in Listing 517, 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 517 more concise by exploring the substitution field. Using the settings in Listing 519 and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=colsep1.yaml
```

then we achieve the output in Listing 518.

The code given in Listing 519 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, [33] for a detailed covering of the topic. With reference to Listing 519, 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 519 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 145** We'll keep working with the file in Listing 515 on the previous page for this example.

Using the YAML in Listing 521, and running the command
7.3 Examples of replacements

Listing 520: colsep.tex using Listing 521

multi-line!

With reference to Listing 521, 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 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 510 on page 126. 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 523, and running the command

Listing 523: multi-line1.yaml

replacements:
- this: |-
  \begin{env}
  1 2 3|arraycolsep=3pt
  4 5 6|arraycolsep=5pt
  \end{env}
  that: 'multi-line!' when: after

We note that, because we have specified when: after, that \latexindent.pl has not found the string specified in Listing 523 within the file in Listing 515 on page 127. As it has looked for the string within Listing 523 after the indentation has been performed. After indentation, the string as written in Listing 523 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,

then the when field is ignored, no indentation is done, and the output is as in Listing 520.

example 146 An important part of the substitution routine is in capture groups.

Assuming that we start with the code in Listing 524, 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 524: displaymath.tex**

before text $$a^2+b^2=4$$ and $$c^2$$

$$
\begin{equation*}
d^2+e^2 = f^2
\end{equation*}
$$
and also $$g^2$$ and some inline math: $h^2$

We use the settings in Listing 526 and run the command

```
cmh:~$ latexindent.pl -r displaymath.tex -l=displaymath1.yaml
```

to receive the output given in Listing 525.

**LISTING 525: displaymath.tex using Listing 526**

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: $h^2$

A few notes about Listing 526:

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 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 6.3 on page 105, which we do in Listing 528; upon running the command

```
cmh:~$ latexindent.pl -r -m displaymath.tex -l=displaymath1.yaml, equation.yaml
```
then we receive the output in Listing 527.
7.3 Examples of replacements

**Listing 527:** displaymath.tex using Listings 526 and 528

```latex
\begin{equation*}
a^2+b^2=4
\end{equation*}
```

and

```latex
\begin{equation*}
c^2
\end{equation*}
```

```latex
\begin{equation*}
d^2+e^2 = f^2
\end{equation*}
```

and also

```latex
\begin{equation*}
g^2
\end{equation*}
```

and some inline math: $h^2$

**example 147** This example is motivated by tex stackexchange question 490086. We begin with the code in Listing 529.

**Listing 529:** phrase.tex

```
phrase 1 phrase 2 phrase 3 phrase 100
phrase 1 phrase 2 phrase 3 phrase 100
phrase 1 phrase 2 phrase 3 phrase 100
phrase 1 phrase 2 phrase 3 phrase 100
```

Our goal is to make the spacing uniform between the phrases. To achieve this, we employ the settings in Listing 531, and run the command

```bash
$ latexindent ∼
  pl -r phrase.tex -l=hspace.yaml
```

which gives the output in Listing 530.

**Listing 530:** phrase.tex using Listing 531

```
phrase 1 phrase 2 phrase 3 phrase 100
phrase 1 phrase 2 phrase 3 phrase 100
phrase 1 phrase 2 phrase 3 phrase 100
phrase 1 phrase 2 phrase 3 phrase 100
```

**Listing 531:** hspace.yaml

```
replacements:
  - substitution: s/\h+/ /sg
```

The \h+ setting in Listing 531 say to replace *at least one horizontal space* with a single space.

**example 148** We begin with the code in Listing 532.
7.3 Examples of replacements

**LISTING 532: references.tex**

equation \eqref{eq:aa} and Figure \ref{fig:bb} and table \ref{tab:cc}

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 534 and running the command

```
$ latexindent -r references.tex -l=reference.yaml
```

which gives the output in Listing 533.

**LISTING 533: references.tex using Listing 534**

\hyperref[equation \ref*{eq:aa}]{equation} and \hyperref[Figure \ref*{fig:bb}]{Figure} and \hyperref[table \ref*{tab:cc}]{table}

**LISTING 534: reference.yaml**

```
replacements:
- substitution: |
  s/(\(equation\)|\(table\)|\(figure\)|\(section\))\(?:eq)?\ref\{(.*?)\}/\hyperref{$1 \ref*{$3}}/sgxi
```

Referencing Listing 534, the | means or, we have used capture groups, together with an example of an optional pattern, (?:eq)?.

**example 149** Let's explore the three replacement mode switches (see Table 4 on page 126) in the context of an example that contains a verbatim code block, Listing 535; we will use the settings in Listing 536.

**LISTING 535: verb1.tex**

```
\begin{myenv}
body of verbatim
\end{myenv}
some verbatim
\begin{verbatim}
body
of
verbatim
text
\end{verbatim}
text
```

```
replacements:
- this: 'body'
  that: 'head'
```

Upon running the following commands,

```
$ latexindent -r verb1.tex -l=verbatim1.yaml -o=+mod1
$ latexindent -rv verb1.tex -l=verbatim1.yaml -o=+-rv-mod1
$ latexindent -rr verb1.tex -l=verbatim1.yaml -o=+-rr-mod1
```
7.3 Examples of replacements

we receive the respective output in Listings 537 to 539

<table>
<thead>
<tr>
<th>Listing 537: verb1-mod1.tex</th>
<th>Listing 538: verb1-rv-mod1.tex</th>
<th>Listing 539: verb1-rr-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{myenv}</td>
<td>\begin{myenv}</td>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>head of verbatim</td>
<td>head of verbatim</td>
<td>head of verbatim</td>
</tr>
<tr>
<td>\end{myenv}</td>
<td>\end{myenv}</td>
<td>\end{myenv}</td>
</tr>
<tr>
<td>some verbatim</td>
<td>some verbatim</td>
<td>some verbatim</td>
</tr>
<tr>
<td>\begin{verbatim}</td>
<td>\begin{verbatim}</td>
<td>\begin{verbatim}</td>
</tr>
<tr>
<td>head of verbatim</td>
<td>head of verbatim</td>
<td>head of verbatim</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>\end{verbatim}</td>
<td>\end{verbatim}</td>
<td>\end{verbatim}</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

We note that:

1. in Listing 537 indentation has been performed, and that the replacements specified in Listing 536 have been performed, even within the verbatim code block;
2. in Listing 538 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 539 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 126.

example 150 Let's explore the \texttt{amalgamate} field from Listing 510 on page 126 in the context of the file specified in Listing 540.

<table>
<thead>
<tr>
<th>Listing 540: amalg1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>one two three</td>
</tr>
</tbody>
</table>

Let’s consider the YAML files given in Listings 541 to 543.

<table>
<thead>
<tr>
<th>Listing 541: amalg1-yaml.yaml</th>
<th>Listing 542: amalg2-yaml.yaml</th>
<th>Listing 543: amalg3-yaml.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>replacements:</td>
<td>replacements:</td>
<td>replacements:</td>
</tr>
<tr>
<td>- this: one</td>
<td>- this: two</td>
<td>- amalgamate: 0</td>
</tr>
<tr>
<td>that: 1</td>
<td>that: 2</td>
<td>that: 3</td>
</tr>
</tbody>
</table>

Upon running the following commands,

```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml
```

we receive the respective output in Listings 544 to 546.

<table>
<thead>
<tr>
<th>Listing 544: amalg1.tex using Listing 541</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 two three</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 545: amalg1.tex using Listings 541 and 542</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 three</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 546: amalg1.tex using Listings 541 to 543</th>
</tr>
</thead>
<tbody>
<tr>
<td>one two 3</td>
</tr>
</tbody>
</table>

We note that:

1. in Listing 544 the replacements from Listing 541 have been used;
2. in Listing 545 the replacements from Listings 541 and 542 have both been used, because
the default value of amalgamate is 1;

3. in Listing 546 only the replacements from Listing 543 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

```bash
$ latexindent.pl --lines 3-7 myfile.tex
$ latexindent.pl -n 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 547.

```
Before the environments
\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}
```

We demonstrate the basic usage using the command

```bash
$ 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 548.
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 to \texttt{latexindent.pl} are equivalent:

```
$ latexindent.pl --lines 3-7 myfile.tex -o=+-mod1
$ latexindent.pl --lines 7-3 myfile.tex -o=+-mod1
```

as \texttt{latexindent.pl} performs a check to put the lowest number first.

\textbf{example 152} You can call the \texttt{lines} switch with only \textit{one number} and in which case only that line will be operated upon. For example

```
$ latexindent.pl --lines 5 myfile.tex
$ latexindent.pl --lines 5-5 myfile.tex
```

instructs \texttt{latexindent.pl} to only operate on line 5; the output is given in Listing 549.

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:

```
$ latexindent.pl --lines 5 myfile.tex
$ latexindent.pl --lines 5-5 myfile.tex
```

\textbf{example 153} If you specify a value outside of the line range of the file then \texttt{latexindent.pl} will ignore the \texttt{lines} argument, detail as such in the log file, and proceed to operate on the entire file.

For example, in the following call

```
$ latexindent.pl --lines 11-13 myfile.tex
```

\texttt{latexindent.pl} will ignore the \texttt{lines} argument, and \textit{operate on the entire file} because List-
ing only has 12 lines.

Similarly, in the call

```bash
$ 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 154** You can specify *multiple line ranges* as in the following

```bash
$ 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 550.

**Listing 550: 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
$ latexindent.pl --lines 3-5,8-10 myfile.tex
$ latexindent.pl --lines 8-10,3-5 myfile.tex
$ latexindent.pl --lines 10-8,3-5 myfile.tex
$ 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 155** 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
$ 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 551.
As previously, the ordering does not matter, and the following calls to `latexindent.pl` are all equivalent:

```bash
$ 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
```

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 156** You can specify negated line ranges by using `!` as in:

```bash
$ latexindent.pl --lines !5-7 myfile.tex -o=+-mod5
```

which instructs `latexindent.pl` to operate upon all of the lines except lines 5 to 7.

In other words, `latexindent.pl` will operate on lines 1 to 4, and 8 to 12, so the following two calls are equivalent:

```bash
$ latexindent.pl --lines !5-7 myfile.tex
$ latexindent.pl --lines 1-4,8-12 myfile.tex
```

The output is given in Listing 552.

### Listing 552: myfile-mod5.tex

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}

**example 157** You can specify multiple negated line ranges such as
\texttt{cmh:~$ latexindent.pl --lines !5-7,!9-10 myfile.tex -o=+-mod6}

which is equivalent to:

\texttt{cmh:~$ latexindent.pl --lines 1-4,8,11-12 myfile.tex -o=+-mod6}

The output is given in Listing 553.

\begin{verbatim}
LISTING 553: myfile-mod6.tex
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}
\end{verbatim}

\textbf{example 158} \ If you specify a line range with anything other than an integer, then \texttt{latexindent.pl} will ignore the \texttt{lines} argument, and \textit{operate on the entire file}.

Sample calls that result in the \texttt{lines} argument being ignored include the following:

\begin{verbatim}
cmh:~$ latexindent.pl --lines 1-x myfile.tex
cmh:~$ latexindent.pl --lines !y-3 myfile.tex
\end{verbatim}

\textbf{example 159} \ We can, of course, use the \texttt{lines} switch in combination with other switches.

For example, let's use with the file in Listing 554.

\begin{verbatim}
LISTING 554: myfile1.tex
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} body \end{two}
7 \end{one}
\end{verbatim}

We can demonstrate interaction with the \texttt{-m} switch (see Section 6 on page 77); in particular, if we use Listing 446 on page 111, Listing 430 on page 109 and Listing 431 on page 109 and run

\begin{verbatim}
cmh:~$ latexindent.pl --lines 6 myfile1.tex -o=+-modi -m -l env-mlb2,env-mlb7,env-mlb8 -o=+-mod1
\end{verbatim}

then we receive the output in Listing 555.
Before the environments
\begin{one}
first block, first line
first block, second line
first block, third line
\end{one}

\begin{two}
body
\end{two}
SECTION 9

Fine tuning

latexindent.pl operates by looking for the code blocks detailed in Table 2 on page 54. The fine tuning of the details of such code blocks is controlled by the fineTuning field, detailed in Listing 556. 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 556: fineTuning

629   fineTuning:
630   environments:
631   name: [a-zA-Z0\*0-9\._]+
632   ifElseFi:
633   name: (?![@]if[a-zA-Z0\*]*\{)@?if[a-zA-Z0\*]*?
634   commands:
635   name: [+a-zA-Z0\*0-9\._:]+?
636   items:
637   canBeFollowedBy: (?:\[\[\]*\]*)|(?:<\[\]*)?
638   keyEqualsValuesBracesBrackets:
639   name: [a-zA-Z0\*0-9\._:]+?
640   follow: (?:(?<!\{)|,|(?:(?<!\[)\[)
641   namedGroupingBracesBrackets:
642   name: [0-9\._a-zA-Z0\*<\]+?
643   follow: \h|\R|\{|\[|\$|\)
644   UnNamedGroupingBracesBrackets:
645   follow: \{|\[|,|&|\)|\(\$
646   arguments:
647   before: (?:[^`\*]+;?,?\/+<\.*?>)
648   between: _|^|^*
649   trailingComments:
650   notPreceededBy: (?<!\)
651   afterComment: .?*
652   modifyLineBreaks:
653   doubleBackSlash: \\(?:\h*\d+[a-zA-Z0\*]+\h*\)
654   comma: ','
655   betterFullStop: |-?
656   (?x)
657   (?: # ignore spaces in the below
658   \.)# # .
659   (?![^\*]+\h*) # not *followed by* a-z
660   )
661   | # OR
662   (?:
663   (?<\[eE]\.[gG]) # not *preceded by*
664   (?:
665   (?:[eE]\.[gG]) # e.g OR E.g OR e.G OR E.G
666   |
```
The fields given in Listing 556 are all regular expressions. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example,[33] for a detailed covering of the topic.

We make the following comments with reference to Listing 556:

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'
   - (d) \* stars
   - (e) 0-9 numbers
   - (f) _ underscores
   - (g) \ backslashes
   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\ha* 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

   [git] • main @ c702424 • 2023-07-02 • ♦ • V3.22.1
5. the `modifyLineBreaks` field refers to fine tuning settings detailed in Section 6 on page 77. In particular:

(a) `betterFullStop` is in relation to the one sentence per line routine, detailed in Section 6.2 on page 93

(b) `doubleBackSlash` is in relation to the `DBSStartsOnOwnLine` and `DBSFinishesWithLineBreak` polyswitches surrounding double backslashes, see Section 6.3.2 on page 115

(c) `comma` is in relation to the `CommaStartsOnOwnLine` and `CommaFinishesWithLineBreak` polyswitches surrounding commas in optional and mandatory arguments; see Table 3 on page 120

It is not obvious from Listing 556, 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 160**

As a demonstration, consider the file given in Listing 557, together with its default output using the command

```
cmh:~$ latexindent.pl finetuning1.tex
```

is given in Listing 558.

<table>
<thead>
<tr>
<th>Listing 557: finetuning1.tex</th>
<th>Listing 558: finetuning1.tex default</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand{ \rule{G -&gt; +H[-G]CL} \rule{H -&gt; -G[+H]CL} \rule{g -&gt; +[-g]cL} \rule{h -&gt; -g[+h]cL} }</td>
<td>\mycommand{ \rule{G -&gt; +H[-G]CL} \rule{H -&gt; -G[+H]CL} \rule{g -&gt; +[-g]cL} \rule{h -&gt; -g[+h]cL} }</td>
</tr>
</tbody>
</table>

It's clear from Listing 558 that the indentation scheme has not worked as expected. We can `fine tune` the indentation scheme by employing the settings given in Listing 560 and running the command

```
cmh:~$ latexindent.pl finetuning1.tex -l=fine-tuning1.yaml
```

and the associated (desired) output is given in Listing 559.

<table>
<thead>
<tr>
<th>Listing 559: finetuning1.tex using Listing 560</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand{ \rule{G -&gt; +H[-G]CL} \rule{H -&gt; -G[+H]CL} \rule{g -&gt; +[-g]cL} \rule{h -&gt; -g[+h]cL} }</td>
</tr>
</tbody>
</table>

**example 161**

Let's have another demonstration; consider the file given in Listing 561, together with its default output using the command

```
cmh:~$ latexindent.pl finetuning2.tex
```
is given in Listing 562.

<table>
<thead>
<tr>
<th>Listing 561: finetuning2.tex</th>
<th>Listing 562: finetuning2.tex default</th>
</tr>
</thead>
<tbody>
<tr>
<td>@misc{ wikilatex,</td>
<td>@misc{ wikilatex,</td>
</tr>
<tr>
<td>author = &quot;{Wikipedia contributors}&quot;},</td>
<td>author = &quot;{Wikipedia contributors}&quot;},</td>
</tr>
<tr>
<td>title = &quot;LaTeX --- {Wikipedia}{,}&quot;},</td>
<td>title = &quot;LaTeX --- {Wikipedia}{,}&quot;},</td>
</tr>
<tr>
<td>note = &quot;[Online; accessed 3-March-2020]&quot; }</td>
<td>note = &quot;[Online; accessed 3-March-2020]&quot; }</td>
</tr>
</tbody>
</table>

It’s clear from Listing 562 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 564 and running the command

```
$ latexindent -l=fine-tuning2.yaml finetuning2.tex
```

and the associated (desired) output is given in Listing 563.

<table>
<thead>
<tr>
<th>Listing 563: finetuning2.tex using Listing 564</th>
</tr>
</thead>
<tbody>
<tr>
<td>@misc{ wikilatex,</td>
</tr>
<tr>
<td>author = &quot;{Wikipedia contributors}&quot;},</td>
</tr>
<tr>
<td>title = &quot;LaTeX --- {Wikipedia}{,}&quot;},</td>
</tr>
<tr>
<td>note = &quot;[Online; accessed 3-March-2020]&quot; }</td>
</tr>
</tbody>
</table>

In particular, note that the settings in Listing 564 specify that NamedGroupingBracesBrackets and UnNamedGroupingBracesBrackets can follow " and that we allow --- between arguments.

**example 162** 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 565 and running the following command

```
$ latexindent -m -y='modifyLineBreaks:oneSentencePerLine:manipulateSentences:␣1,␣ modifyLineBreaks:oneSentencePerLine:sentencesBeginWith:a-z:␣1,␣ fineTuning:modifyLineBreaks:betterFullStop:␣ "(?:\.|;|:(?![a-z]))|(?:(?!:(?:e\.|g)|(?:i\.|e)|(?:etc))(?!:([a-z]|A-Z)\.|issue-243.txt)\s|\-+mod1)'
```

gives the output shown in Listing 566.

<table>
<thead>
<tr>
<th>Listing 565: finetuning3.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>We go; you see: this sentence \cite{tex:stackexchange} finishes here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 566: finetuning3.tex using -y switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>We go; you see: this sentence \cite{tex:stackexchange} finishes here.</td>
</tr>
</tbody>
</table>

**example 163** We can tweak the fineTuning for how trailing comments are classified. For motivation, let’s consider the code given in Listing 567

```
\cite{tex:stackexchange}
```
We will compare the settings given in Listings 568 and 569.

---

**Listing 568: href1.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: -1
  blocksEndBefore:
    verbatim: 0
  blocksFollow:
    verbatim: 0
removeTrailingWhitespace:
  beforeProcessing: 1
```

**Listing 569: href2.yaml**

```yaml
fineTuning:
  trailingComments:
    notPreceededBy:
      '(?:(?<!Handbook)(?<!for)(?<!Spoken))'
modifyLineBreaks:
  textWrapOptions:
    columns: -1
  blocksEndBefore:
    verbatim: 0
  blocksFollow:
    verbatim: 0
removeTrailingWhitespace:
  beforeProcessing: 1
```

---

Upon running the following commands

```bash
$ 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 570 and 571.

---

**Listing 570: finetuning4.tex using Listing 568**

some before text \href{Handbook\%20for\%30Spoken\%40document.pdf}{my document}

---

**Listing 571: finetuning4.tex using Listing 569**

some before text \href{Handbook\%20for\%30Spoken\%40document.pdf}{my document} some after text

We note that in:

- Listing 570 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 571 has fine-tuned the trailing comment matching, and says that % cannot be immediately preceded 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.

---

**example 164**

Another approach to this situation, which does not use fineTuning, is to use noIndentBlock which we discussed in Listing 44 on page 30; using the settings in Listing 572 and running the command

```bash
$ latexindent.pl -m finetuning4.tex -o=+-mod3 -l=href3
```

then we receive the same output given in Listing 571.
With reference to the `body` field in Listing 572, 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 165**

We can use the `fineTuning` field to assist in the formatting of bibliography files.

Starting with the file in Listing 573 and running the command

```
cmh:~$ latexindent.pl bib1.bib -o=-mod1
```

gives the output in Listing 574.

Let's assume that we would like to format the output so as to align the `=` symbols. Using the settings in Listing 576 and running the command

```
cmh:~$ latexindent.pl bib1.bib -l bibsettings1.yaml -o=-mod2
```

gives the output in Listing 575.

Some notes about Listing 576:
• we have populated the lookForAlignDelims field with the online command, and have used the delimiterRegEx, discussed in Section 5.5.4 on page 43;

• 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 166** We can build upon Listing 576 for slightly more complicated bibliography files.

Starting with the file in Listing 577 and running the command

```bash
$ latexindent .pl bib2.bib -l bibsettings1.yaml -o=+-mod1
```

gives the output in Listing 578.

**Listing 577: bib2.bib**

```yaml
@online{cmh:videodemo,
title="Videodemonstrationofpl.latexindentonyoutube",
url="https://www.youtube.com/watch?v=wo38aaH2F4E&spfreload=10",
urldate={2017-02-21},
}
```

**Listing 578: bib2-mod1.bib**

```yaml
@online{cmh:videodemo,
title = "Videodemonstrationofpl.latexindentonyoutube",
url = "https://www.youtube.com/watch?v=wo38aaH2F4E&spfreload = 10",
urldate = {2017-02-21},
}
```

The output in Listing 578 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 579.

**Listing 579: bibsettings2.yaml**

```yaml
lookForAlignDelims:
  online:
    delimiterRegEx: '(?<!v)(?<!spfreload)(=)'
```

Upon running the command

```bash
$ latexindent .pl bib2.bib -l bibsettings1.yaml,bibsettings2.yaml -o=+-mod2
```

we receive the *desired* output in Listing 580.

**Listing 580: bib2-mod2.bib**

```yaml
@online{cmh:videodemo,
title = "Videodemonstrationofpl.latexindentonyoutube",
url = "https://www.youtube.com/watch?v=wo38aaH2F4E&spfreload = 10",
urldate = {2017-02-21},
}
```

With reference to Listing 579 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.
We can use the fineTuning settings to tweak how latexindent.pl finds trailing comments.

We begin with the file in Listing 581

```
\chapter{chapter text} \% 123
chapter text
\section{section text} \% 456
section text
\% end
\% end
```

Using the settings in Listing 583 and running the command

```
$ latexindent.pl finetuning5.tex -l=fine-tuning3.yaml
```

gives the output in Listing 582.

```
\chapter{chapter text} \% 123
chapter text
\section{section text} \% 456
section text
\% end
\% end
```

The settings in Listing 583 detail that trailing comments can not be followed by a single space, and then the text ‘end’. This means that the specialBeginEnd routine will be able to find the pattern \% end as the end part. The trailing comments 123 and 456 are still treated as trailing comments.
There are a number of known limitations of the script, and almost certainly quite a few that are unknown! The known issues include:

**Multicolumn alignment** when working with code blocks in which multicolumn commands overlap, the algorithm can fail; see Listing 72 on page 36.

**TextWrap after** when operating with indentRules (see Section 5.8 on page 53) may not always cooperate with one another; if you have a specific example that does not work, please report it to [34].

**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 124); 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 latexindent on any file; if you don’t specify an extension, then the extensions that you specify in fileExtensionPreference (see Listing 36 on page 27) will be consulted. If you find a case in which the script struggles, please feel free to report it at [34], and in the meantime, consider using a noIndentBlock (see page 30).

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 [34]; otherwise, feel free to find me on the [http://tex.stackexchange.com/users/6621/cmhughes](http://tex.stackexchange.com/users/6621/cmhughes).
References

11.1 perl-related links

[34] Text::Tabs Perl module. URL: http://search.cpan.org/~muir/Text-Tabs+Wrap-2013.0523/lib.old/Text/Tabs.pm (visited on 07/06/2017).

11.2 conda-related links


11.3 VScode-related links


11.4 Other links

[38] latexindent.pl ghcr (GitHub Container Repository) location. URL: https://github.com/cmhughes?tab=packages (visited on 06/12/2022).


11.5 Contributors (in chronological order)


SECTION A

Required Perl modules

If you intend to use latexindent.pl and not one of the supplied standalone executable files (latexindent.exe is available for Windows users without Perl, see Section 3.1.2), then you will need a few standard Perl modules.

If you can run the minimum code in Listing 584 as in

```
cmh:~$ 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.

```
#!usr/bin/perl

use strict; # |
use warnings; # |
use Encode; # |
use Getopt::Long; # |
use Data::Dumper; # these modules are
use List::Util qw(max); # generally part
use PerlIO::encoding; # of a default perl distribution
use open ':std', ':encoding(UTF-8)';# |
use Text::Wrap; # |
use Text::Tabs; # |
use FindBin; # |
use File::Copy; # |
use File::Basename; # |
use File::HomeDir; # <--- typically requires install via cpanm
use YAML::Tiny; # <--- typically requires install via cpanm

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 584 will vary depending on your operating system and Perl distribution.

A.2.1 Linux

A.2.1.1 perlbrew

Linux users may be interested in exploring Perlbrew [40]; an example installation would be:

```
cmh:~$ sudo apt-get install perlbrew
cmh:~$ perlbrew init
cmh:~$ perlbrew install perl-5.34.0
cmh:~$ perlbrew switch perl-5.34.0
cmh:~$ sudo apt-get install curl
```
```
cmh:~$ curl -L http://cpanmin.us | perl - App::cpanminus
```
```
cmh:~$ cpanm YAML::Tiny
```
```
cmh:~$ cpanm File::HomeDir
```

A.2.1.2 Ubuntu/Debian

For other distributions, the Ubuntu/Debian approach may work as follows

```
cmh:~$ sudo apt install perl
```
```
cmh:~$ sudo cpan -i App::cpanminus
```
```
cmh:~$ sudo cpanm YAML::Tiny
```
```
cmh:~$ sudo cpanm File::HomeDir
```

or else by running, for example,

```
cmh:~$ 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

```
cmh:~$ sudo apt install texlive
```
```
cmh:~$ sudo apt install texlive-latex-recommended
```

may need the following additional command to work with latexindent.pl

```
cmh:~$ sudo apt install texlive-extra-utils
```

A.2.1.4 Ubuntu: users without perl

latexindent-linux is a standalone executable for Ubuntu Linux (and therefore does not require a Perl distribution) and caches copies of the Perl modules onto your system. It is available from [34].

A.2.1.5 Arch-based distributions

First install the dependencies

```
cmh:~$ sudo pacman -S perl cpanminus
```

In addition, install perl-file-homedir from AUR, using your AUR helper of choice,
then run the latexindent-module-installer.pl file located at helper-scripts/

A.2.1.6 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 585; thanks to [12] for providing these details.

```bash
LISTING 585: alpine-install.sh

# Installing perl
apk --no-cache add miniperl perl-utils

# Installing incompatible latexindent perl dependencies via apk
apk --no-cache add \n   perl-log-dispatch \n   perl-namespace-autoclean \n   perl-specio \n   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
```

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:

```bash
cmh:~$ brew install perl
cmh:~$ brew install cpanm
cmh:~$
cmh:~$ cpanm YAML::Tiny
cmh:~$ cpanm File::HomeDir
```

Alternatively,

```bash
cmh:~$ brew install latexindent
```

latexindent-macos is a standalone executable for macOS (and therefore does not require a Perl distribution) and caches copies of the Perl modules onto your system. It is available from [34].

A.2.3 Windows

Strawberry Perl users on Windows might use CPAN client. All of the modules are readily available on CPAN [30]. 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
A.3 The GCString switch

If you find that the `lookForAlignDelims` (as in Section 5.5) does not work correctly for your language, then you may wish to use the `Unicode::GCString` module.

This can be loaded by calling `latexindent.pl` with the `GCString` switch as in

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

In this case, you will need to have the `Unicode::GCString` installed in your `perl` distribution by using, for example,

```
cmh:~$ cpanm Unicode::GCString
```

Note: this switch does *nothing* for `latexindent.exe` which loads the module by default. Users of `latexindent.exe` should not see any difference in behaviour whether they use this switch or not, as `latexindent.exe` loads the `Unicode::GCString` module.
SECTION B

Updating the path variable

latexindent.pl has a few scripts (available at [34]) that can update the path variables. Thank you to [6] for this feature. If you're on a Linux or Mac machine, then you'll want CMakeLists.txt from [34].

B.1 Add to path for Linux

To add latexindent.pl to the path for Linux, follow these steps:

1. download latexindent.pl and its associated modules, defaultSettings.yaml, to your chosen directory from [34];

2. within your directory, create a directory called path-helper-files and download CMakeLists.txt and cmake_uninstall.cmake.in from [34]/path-helper-files to this directory;

3. run

```bash
cmh:~$ ls /usr/local/bin
```
to see what is currently in there;

4. run the following commands

```bash
cmh:~$ sudo apt-get update
cmh:~$ sudo apt-get install --no-install-recommends cmake make # or any other generator
cmh:~$ mkdir build && cd build
cmh:~$ cmake ../path-helper-files
cmh:~$ sudo make install
```

5. run

```bash
cmh:~$ ls /usr/local/bin
```
again to check that latexindent.pl, its modules and defaultSettings.yaml have been added.

To remove the files, run

```bash
cmh:~$ sudo make uninstall
```

B.2 Add to path for Windows

To add latexindent.exe to the path for Windows, follow these steps:

1. download latexindent.exe, defaultSettings.yaml, add-to-path.bat from [34] to your chosen directory;

2. open a command prompt and run the following command to see what is currently in your %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.
SECTION C

Batches of files

You can instruct `latexindent.pl` to operate on multiple files. For example, the following calls are all valid:

```
cmh:~$ latexindent.pl myfile1.tex
cmh:~$ latexindent.pl myfile1.tex myfile2.tex
cmh:~$ latexindent.pl myfile*.tex
```

We note the following features of the script in relation to the switches detailed in Section 3.

C.1 location of indent.log
If the `-c` switch is not active, then `indent.log` goes to the directory of the final file called.
If the `-c` switch is active, then `indent.log` goes to the specified directory.

C.2 interaction with `-w` switch
If the `-w` switch is active, as in

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

then files will be overwritten individually. Back-up files can be re-directed via the `-c` switch.

C.3 interaction with `-o` switch
If `latexindent.pl` is called using the `-o` switch as in

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

and there are multiple files to operate upon, then the `-o` switch is ignored because there is only one output file specified.

More generally, if the `-o` switch does not have a `+` symbol at the beginning, then the `-o` switch will be ignored, and is turned it off.

For example

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

will work fine because each `.tex` file will output to `<basename>myfile.tex`

Similarly,

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

will work because the 'existence check/incrementation' routine will be applied.

C.4 interaction with lines switch
This behaves as expected by attempting to operate on the line numbers specified for each file. See the examples in Section 8.
C.5  interaction with check switches

The exit codes for \texttt{latexindent.pl} are given in Table 1 on page 22.

When operating on multiple files with the check switch active, as in

\begin{verbatim}
~$ latexindent.pl myfile*.tex --check
\end{verbatim}

then

\begin{itemize}
  \item exit code 0 means that the text from \textit{none} of the files has been changed;
  \item exit code 1 means that the text from \textit{at least one} of the files been file changed.
\end{itemize}

The interaction with \texttt{checkv} switch is as in the check switch, but with verbose output.

C.6  when a file does not exist

What happens if one of the files can not be operated upon?

\begin{itemize}
  \item if at least one of the files does not exist and \texttt{latexindent.pl} has been called to act upon multiple files, then the exit code is 3; note that \texttt{latexindent.pl} will try to operate on each file that it is called upon, and will not exit with a fatal message in this case;
  \item if at least one of the files can not be read and \texttt{latexindent.pl} has been called to act upon multiple files, then the exit code is 4; note that \texttt{latexindent.pl} will try to operate on each file that it is called upon, and will not exit with a fatal message in this case;
  \item if \texttt{latexindent.pl} has been told to operate on multiple files, and some do not exist and some cannot be read, then the exit code will be either 3 or 4, depending upon which it scenario it encountered most recently.
\end{itemize}
latexindent.pl ships with latexindent-yaml-schema.json which might help you when constructing your YAML files.

D.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 [34], save it in whichever directory you would like, noting it for reference;

2. following the instructions from [35], for example, you should install the VSCode YAML extension [48];

3. set up your settings.json file using the directory you saved the file by adapting Listing 586; on my Ubuntu laptop this file lives at /home/cmhughes/.config/Code/User/settings.json.

```json
Listing 586: settings.json
{
  "yaml.schemas": {
    "/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 587.

```json
Listing 587: settings-alt.json
{
  "yaml.schemas": {
    "/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 588 may work.

```json
Listing 588: settings-alt1.json
{
  "yaml.schemas": {
    "/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 E**

Using conda

If you use conda you'll only need

```
$ 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 [29] or from [28].

### E.1 Sample conda installation on Ubuntu

On Ubuntu I followed the 64-bit installation instructions at [36] and then I ran the following commands:

```
$ conda create -n latexindent.pl
$ conda activate latexindent.pl
$ conda install latexindent.pl -c conda-forge
$ conda info --envs
$ conda list
$ conda run latexindent.pl -vv
```

I found the details given at [43] to be helpful.
SECTION F

Using docker

If you use docker you’ll only need

```
cmh:~$ docker pull ghcr.io/cmhughes/latexindent.pl
```

This will download the image packed latexindent’s executable and its all dependencies. Thank you to [19] for contributing this feature; see also [38]. For reference, ghcr stands for GitHub Container Repository.

F.1 Sample docker installation on Ubuntu

To pull the image and show latexindent's help on Ubuntu:

```
# setup docker if not already installed
if ! command -v docker &> /dev/null; then
    sudo apt install docker.io -y
    sudo groupadd docker
    sudo gpasswd -a "$USER" docker
    sudo systemctl restart docker
fi

# download image and execute
docker pull ghcr.io/cmhughes/latexindent.pl
docker run ghcr.io/cmhughes/latexindent.pl -h
```

F.2 How to format on Docker

When you use latexindent with the docker image, you have to mount target tex file like this:

```
cmh:~$ docker run -v /path/to/local/myfile.tex:/myfile.tex
ghcr.io/cmhughes/latexindent.pl -s -w myfile.tex
```
Users of `.git` may be interested in exploring the `pre-commit` tool [42], which is supported by `latexindent.pl`. Thank you to [20] for contributing this feature, and to [21] for their contribution to it.

To use the `pre-commit` tool, you will need to install `pre-commit`; sample instructions for Ubuntu are given in appendix G.1. Once installed, there are two ways to use `pre-commit`: using CPAN or using conda, detailed in appendix G.3 and appendix G.4 respectively.

### G.1 Sample pre-commit installation on Ubuntu

On Ubuntu I ran the following command:

```
cmh:~$ python3 -m pip install pre-commit
```

I then updated my path via `.bashrc` so that it includes the line in Listing 590.

```
export PATH=$PATH:/home/cmhughes/.local/bin
```

### G.2 pre-commit defaults

The default values that are employed by `pre-commit` are shown in Listing 591.

```
- id: latexindent
  name: latexindent.pl
  description: Run latexindent.pl (get dependencies using CPAN)
  minimum_pre_commit_version: 2.1.0
  entry: latexindent.pl
  args: ["--overwriteIfDifferent", "--silent", "--local"]
  language: perl
  types: [tex]
- id: latexindent-conda
  name: latexindent.pl
  description: Run latexindent.pl (get dependencies using Conda)
  minimum_pre_commit_version: 2.1.0
  entry: latexindent.pl
  args: ["--overwriteIfDifferent", "--silent", "--local"]
  language: conda
  types: [tex]
- id: latexindent-docker
  name: latexindent.pl
  description: Run latexindent.pl (get dependencies using Docker)
  minimum_pre_commit_version: 2.1.0
  entry: ghcr.io/cmhughes/latexindent.pl
  language: docker_image
  types: [tex]
  args: ["--overwriteIfDifferent", "--silent", "--local"]
```
In particular, the decision has deliberately been made (in collaboration with [21]) to have the default to employ the following switches: overwriteIfDifferent, silent, local; this is detailed in the lines that specify args in Listing 591.

**Warning!**

Users of pre-commit will, by default, have the overwriteIfDifferent switch employed. It is assumed that such users have version control in place, and are intending to overwrite their files.

### G.3 pre-commit using CPAN

To use latexindent.pl with pre-commit, create the file `.pre-commit-config.yaml` given in Listing 592 in your git-repository.

#### Listing 592: `.pre-commit-config.yaml (cpa)`

```yaml
- repo: https://github.com/cmhughes/latexindent.pl
  rev: V3.22.1
  hooks:
    - id: latexindent
      args: [-s]
```

Once created, you should then be able to run the following command:

```bash
cmh:~$ pre-commit run --all-files
```

A few notes about Listing 592:

- the settings given in Listing 592 instruct pre-commit to use CPAN to get dependencies;
- this requires pre-commit and perl to be installed on your system;
- the args lists selected command-line options; the settings in Listing 592 are equivalent to calling

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

for each .tex file in your repository;

- to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 592 so that args: [-s, -w].

Naturally you can add options, or omit -s and -w, according to your preference.

### G.4 pre-commit using conda

You can also rely on conda (detailed in appendix E) instead of CPAN for all dependencies, including latexindent.pl itself.

#### Listing 593: `.pre-commit-config.yaml (conda)`

```yaml
- repo: https://github.com/cmhughes/latexindent.pl
  rev: V3.22.1
  hooks:
    - id: latexindent-conda
      args: [-s]
```

Once created, you should then be able to run the following command:

```bash
cmh:~$ pre-commit run --all-files
```

A few notes about Listing 592:
• the settings given in Listing 593 instruct pre-commit to use conda to get dependencies;
• this requires pre-commit and conda to be installed on your system;
• the args lists selected command-line options; the settings in Listing 592 are equivalent to calling

```
$ conda run latexindent.pl -s myfile.tex
```

for each .tex file in your repository;
• to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 592 so that args: [-s, -w].

G.5 pre-commit using docker

You can also rely on docker (detailed in appendix F) instead of CPAN for all dependencies, including latexindent.pl itself.

```
<table>
<thead>
<tr>
<th>Listing 594: .pre-commit-config.yaml (docker)</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo: <a href="https://github.com/cmhughes/latexindent.pl">https://github.com/cmhughes/latexindent.pl</a></td>
</tr>
<tr>
<td>rev: V3.22.1</td>
</tr>
<tr>
<td>hooks:</td>
</tr>
<tr>
<td>- id: latexindent-docker</td>
</tr>
<tr>
<td>args: [-s]</td>
</tr>
</tbody>
</table>
```

Once created, you should then be able to run the following command:

```
$ pre-commit run --all-files
```

A few notes about Listing 592:
• the settings given in Listing 594 instruct pre-commit to use docker to get dependencies;
• this requires pre-commit and docker to be installed on your system;
• the args lists selected command-line options; the settings in Listing 592 are equivalent to calling

```
$ docker run -v /path/to/myfile.tex:/myfile.tex ghcr.io/cmhughes/latexindent.pl -s myfile.tex
```

for each .tex file in your repository;
• to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 592 so that args: [-s, -w].

G.6 pre-commit example using -l, -m switches

Let's consider a small example, with local latexindent.pl settings in .latexindent.yaml.

**example 168** We use the local settings given in Listing 595.

```
<table>
<thead>
<tr>
<th>Listing 595: .latexindent.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>onlyOneBackUp: 1</td>
</tr>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine: 1</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
</tbody>
</table>
```

and .pre-commit-config.yaml as in Listing 596:
Now running

```shell
cmh:~$ pre-commit run --all-files
```

is equivalent to running

```shell
cmh:~$ latexindent.pl -l -m -s -w myfile.tex
```

for each .tex file in your repository.

A few notes about Listing 596:

- the `-l` option was added to use the local `.latexindent.yaml` (where it was specified to only create one back-up file, as `git` typically takes care of this when you use `pre-commit`);
- `-m` to modify line breaks; in addition to `-s` to suppress command-line output, and `-w` to format files in place.
This section describes the possible locations for the main configuration file, discussed in Section 4. Thank you to [22] for this contribution.

The possible locations of `indentconfig.yaml` are read one after the other, and reading stops when a valid file is found in one of the paths.

Before stating the list, we give summarise in Table 5.

<table>
<thead>
<tr>
<th>environment variable</th>
<th>type</th>
<th>Linux</th>
<th>macOS</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATEXINDENT_CONFIG</td>
<td>full path to file</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>XDG_CONFIG_HOME</td>
<td>directory path</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>LOCALAPPDATA</td>
<td>directory path</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

The following list shows the checked options and is sorted by their respective priority. It uses capitalized and with a dollar symbol prefixed names (e.g. `$LATEXINDENT_CONFIG`) to symbolize environment variables. In addition to that the variable name `$homeDir` is used to symbolize your home directory.

1. The value of the environment variable `$LATEXINDENT_CONFIG` is treated as highest priority source for the path to the configuration file.
2. The next options are dependent on your operating system:
   - Linux:
     (a) The file at `$XDG_CONFIG_HOME/latexindent/indentconfig.yaml`
     (b) The file at `$homeDir/.config/latexindent/indentconfig.yaml`
   - Windows:
     (a) The file at `$LOCALAPPDATA\latexindent\indentconfig.yaml`
     (b) The file at `$homeDir\AppData\Local\latexindent\indentconfig.yaml`
   - Mac:
     (a) The file at `$homeDir/Library/Preferences/latexindent/indentconfig.yaml`
3. The file at `$homeDir/indentconfig.yaml`
4. The file at `$homeDir/.indentconfig.yaml`

H.1 Why to change the configuration location

This is mostly a question about what you prefer, some like to put all their configuration files in their home directory (see `$homeDir` above), whilst some like to sort their configuration. And if you don’t care about it, you can just continue using the same defaults.
H.2 How to change the configuration location

This depends on your preferred location, if, for example, you would like to set a custom location, you would have to change the $LATEXINDENT_CONFIG environment variable.

Although the following example only covers $LATEXINDENT_CONFIG, the same process can be applied to $XDG_CONFIG_HOME or $LOCALAPPDATA because both are environment variables. You just have to change the path to your chosen configuration directory (e.g. $homeDir/.config or $homeDir/AppData\Local on Linux or Windows respectively)

H.2.1 Linux

To change $LATEXINDENT_CONFIG on Linux you can run the following command in a terminal after changing the path:

```
cmh:~$ echo 'export LATEXINDENT_CONFIG="/home/cmh/latexindent-config.yaml"' >> ~/.profile
```

Context: This command adds the given line to your .profile file (which is commonly stored in $HOME/.profile). All commands in this file are run after login, so the environment variable will be set after your next login.

You can check the value of $LATEXINDENT_CONFIG by typing

```
cmh:~$ echo $LATEXINDENT_CONFIG
```

Linux users interested in $XDG_CONFIG_HOME can explore variations of the following commands

```
cmh:~$ echo $XDG_CONFIG_HOME
cmh:~$ echo ${XDG_CONFIG_HOME:=$HOME/.config}
cmh:~$ mkdir /home/cmh/.config/
```

H.2.2 Windows

To change $LATEXINDENT_CONFIG on Windows you can run the following command in powershell.exe after changing the path:

```
C:\Users\cmh> [Environment]::SetEnvironmentVariable
C:\Users\cmh> ("LATEXINDENT_CONFIG", "$\your\config\path", "User")
```

This sets the environment variable for every user session.

H.2.3 Mac

To change $LATEXINDENT_CONFIG on macOS you can run the following command in a terminal after changing the path:

```
cmh:~$ echo 'export LATEXINDENT_CONFIG="/your/config/path"' >> ~/.profile
```

Context: This command adds the line to your .profile file (which is commonly stored in $HOME/.profile). All commands in this file are run after login, so the environment variable will be set after your next login.
Listing 37 on page 28 describes the options for customising the information given to the log file, and we provide a few demonstrations here.

**example 169** Let's say that we start with the code given in Listing 597, and the settings specified in Listing 598.

**LISTING 597: simple.tex**

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

**LISTING 598: logfile-prefs1.yaml**

```
logFilePreferences:
  showDecorationStartCodeBlockTrace: "+++++
  showDecorationFinishCodeBlockTrace: "-----"
```

If we run the following command (noting that `-t` is active)

```
cmh:~$ latexindent.pl -t -l=logfile-prefs1.yaml simple.tex
```

then on inspection of `indent.log` we will find the snippet given in Listing 599.

**LISTING 599: indent.log**

```
+++++
TRACE: environment found: myenv
No ancestors found for myenv
Storing settings for myenvvironments
indentRulesGlobal specified (0) for environments, ...
Using defaultIndent for myenv
Putting linebreak after replacementText for myenv
looking for COMMANDS and key = {value}
TRACE: Searching for commands with optional and/or mandatory arguments AND key = {value}
     looking for SPECIAL begin/end
TRACE: Searching myenv for special begin/end (see specialBeginEnd)
TRACE: Searching myenv for optional and mandatory arguments
     ... no arguments found
-----
```

Notice that the information given about `myenv` is ‘framed’ using `+++++` and `-----` respectively.
SECTION J

Encoding indentconfig.yaml

In relation to Section 4 on page 23, Windows users that encounter encoding issues with `indentconfig.yaml`, may wish to run the following command in either `cmd.exe` or `powershell.exe`:

```
C:\Users\cmh>chcp
```

They may receive the following result

```
C:\Users\cmh>Active code page: 936
```

and can then use the settings given in Listing 600 within their `indentconfig.yaml`, where 936 is the result of the `chcp` command.

```
listing 600: encoding demonstration for indentconfig.yaml
encoding: cp936
```

SECTION K

dos2unix linebreak adjustment

If you use \texttt{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 \texttt{dos2unixlinebreaks} to 1 and employing, for example, the following command.

\texttt{cmh:~$ latexindent.pl \ -y\="dos2unixlinebreaks:1"\ myfile.tex}

See [49] for further details.
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,

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

whereas in Version 3.0 you would run any of the following, for example,

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

```

noting that the `output` file is given next to the `-o` switch.

The fields given in Listing 601 are obsolete from Version 3.0 onwards.

```
<table>
<thead>
<tr>
<th>Listing 601: 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 602 and 603

```
<table>
<thead>
<tr>
<th>Listing 602: indentAfterThisHeading in Version 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indent: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Listing 603: indentAfterThisHeading in Version 3.0</th>
</tr>
</thead>
<tbody>
<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 604; as of Version 3.0, you would write YAML as in Listing 605 or, if you're using `-m` switch, Listing 606.
LISTING 604: noAdditionalIndent in Version 2.2

noAdditionalIndent:
\[: 0
\]: 0

LISTING 605: noAdditionalIndent for displayMath in Version 3.0

specialBeginEnd:
displayMath:
  begin: '```['
  end: '```]
  lookForThis: 0

LISTING 606: noAdditionalIndent for displayMath in Version 3.0

noAdditionalIndent:
displayMath: 1

End
Index

— B —
backup files
  cycle through, 25
  extension settings, 24
  maximum number of backup files, 25
  number of backup files, 25
  overwrite switch, -w, 12
  overwritelfDifferent switch, -wd, 12
bibliography files, 139

— C —
capturing parenthesis (regex), 40
comments
  text wrap, 87, 101
conda, 11, 143, 152, 155
contributors, 143
cpAn, 144, 155

— D —
delimiters, 113
  advanced settings, 31
  advanced settings of lookForAlignDelims, 30
  ampersand &, 31
  default settings of lookForAlignDelims, 31
  delimiter justification (left or right), 40
delimiterRegEx, 40, 139
dontMeasure feature, 38
double backslash demonstration, 37
lookForAlignDelims, 31
poly-switches for double backslash, 111
spacing demonstration, 32
  with specialBeginEnd and the -m switch, 113
  within specialBeginEnd blocks, 47
docker, 11, 153, 156

— E —
exit code, 18
  summary, 18

— G —
git, 155, 156

— I —
indentation
  customising indentation per-code block, 50
  customising per-name, 50
default, 15
defaultIndent description, 30
defaultIndent using -y switch, 15
defaultIndent using YAML file, 20
  maximum indetation, 49
  no additional indent, 50
  no additional indent global, 50
  removing indentation per-code block, 50
  summary, 67

— J —
json
  schema for YAML files, 151
  VSCode, 151

— L —
latexindent-linux, 11, 144
latexindent-macos, 11, 145
latexindent.exe, 11
linebreaks
  summary of poly-switches, 113
  linux, 11, 144

— M —
macOS, 11, 145
MiKTeX, 11, 143
modifying linebreaks
  at the beginning of a code block, 102
  at the end of a code block, 104
  by using one sentence per line, 89
  surrounding double backslash, 111
  using poly-switches, 101

— P —
perl
  Unicode GCString module, 146
poly-switches
  adding comments and then line breaks: set to 2, 103, 105
  adding blank lines (again!): set to 4, 104
  adding blank lines: set to 3, 103
  adding blank lines (again!): set to 4, 106
  adding blank lines: set to 3, 105
  adding line breaks: set to 1, 102, 104
  blank lines, 109
  conflicting switches, 117
  conflicting partnering, 116
  conflicting switches, 118
  default values, 102
  definition, 101
  double backslash, 113
  environment global example, 102
  environment per-code block example, 102
  for double back slash (delimiters), 111
  for double backslash (delimiters), 113
  for double backslash (delimiters), 112, 114
off by default: set to 0, 101
removing line breaks: set to -1, 107
summary of all poly-switches, 114
values, 101
visualisation: ♠, ♥, ♦, ♣, 102

— R —
regular expressions
capturing parenthesis, 40
class demonstration, 139
delimeter regex at \= or \\>, 40
delimeter regex at only \\>, 40
dontMeasure feature, cell, 38
dontMeasure feature, row, 39
horizontal space \h, 98, 126
keyEqualsValuesBracesBrackets, 134
lowercase alph a-z, 38, 94
NamedGroupingBracesBrackets, 134
substitution field, arraycolsep, 122
substitution field, equation, 124
UnNamedGroupingBracesBrackets, 134
uppercase alph A-Z, 92, 98

— S —
sentences
begin with, 92

begin with, 93
comments, 101
do with, 92, 94
follow, 92
indenting, 97
one sentence per line, 89
oneSentencePerLine, 89
removing sentence line breaks, 91
text wrap, 101
text wrapping, 97

— specialBeginEnd —
alignment at delimiter, 47
combined with lookForAlignDelims, 47
default settings, 43
delimiterRegEx, 47
delimiterRegEx tweaked, 47
double backslash poly-switch demonstration, 113
IFElSiFi example, 45
indentRules example, 62
indentRulesGlobal, 67
introduction, 43
lookForAlignDelims, 113
middle, 45
noAdditionalIndent, 62
noAdditionalIndentGlobal, 67
poly-switch summary, 114
searching for special before commands, 44
specifying as verbatim, 46
tikz example, 47
update to displaymath V3.0, 163

— switches —
–GCString, 18
–GCString demonstration, 146
-c, –crft definition and details, 16
-d, –onlydefault definition and details, 16
-g, –logfile definition and details, 16
-h, –help definition and details, 12
-k, –check definition and details, 17
-ks, –checkv definition and details, 18
-l demonstration, 112, 114
-l demonstration, 22, 32, 38, 47, 49, 56, 57,
98, 122–124, 136, 137
-l demonstration, 21, 38–41, 45–49, 52–64,
69, 70, 72, 88, 89, 91–97, 99, 103–106,
-l demonstration, 46
-l in relation to other settings, 22
-l, –local definition and details, 14
-lines demonstration, 128
-lines demonstration, negation, 131
-m demonstration, 104, 106, 108
-m demonstration, 89, 93, 95, 116, 117
-m demonstration, 74, 88, 91, 92, 94–99, 103,
105, 109–114, 117–119, 124
-m, –modifylinebreaks definition and details, 16
-m, –modifylinebreaks definition and details, 16
-n, –lines definition and details, 18
-o demonstration, 37
-o demonstration, 41, 47, 88, 89, 126, 163
-o, –output definition and details, 13
-r demonstration, 124
-r demonstration, 120–127
-r, –replacement definition and details, 17
-rr demonstration, 126
-rr demonstration, 123
-rr, –onlyreplacement definition and details, 17
-rv demonstration, 126
-rv, –replacementrespectverb definition and details, 17
-s, –silent definition and details, 14
-sl, –screenlog definition and details, 16
-t, –trace definition and details, 14
-tt, –tttrace definition and details, 14
-v, –version definition and details, 12
-vv, –vversion definition and details, 12
-w, –overwrite definition and details, 12
-wd, –overwritelfDifferent definition and details, 12
-y demonstration, 22, 37, 98
-y, –yaml definition and details, 15

— T —
TeXLive, 11, 144, 145
text wrap
blocksFollow, 77
comments, 78
headings, 77
other, 79, 80, 83
comments, 87
text wrap
blocksBeginWith, 81
blocksEndBefore, 83
comments, 101
quick start, 76
setting columns to -1, 76

— V —
verbatim
cmds, 26
cmds, comparison with -r and -rr switches, 126
environments, 26
in relation to oneSentencePerLine, 95
noIndentBlock, 27
poly-switch summary, 114
rv, replacementrespectverb switch, 120
rv, replacementrespectverb switch, 17
specialBeginEnd, 46
verbatimEnvironments demonstration (-l switch), 22
verbatimEnvironments demonstration (-y switch), 22
VSCode, 143, 151

— W —
warning
amalgamate field, 72
be sure to test before use, 2
capture groups, 136
capturing parenthesis for lookForAlignDelims,
changing huge (textwrap), 88
editing YAML files, 21
fine tuning, 134
pre-commit, 155
the m switch, 74
Windows, 11