Introduction to LaTeX

A Beginner's Guide

This is a simple guide that shall serve as an introduction to LaTeX (ˈlɑːtɛk or ˈleɪtɛk). Test paper.

**Please note that …*

1. What is LaTeX?

Add text here.

2. Structure of a LaTeX-Document

A LaTeX-document consists of two main parts: 1.) the preamble and 2.) the body. The preamble is where the fundamentals of the document are set up, including the type of document, format and packages (see XXX). It may also contain metadata such as the the author, date or title. The preamble is initiated by the command *\documentclass{}* and ends with the beginning of the main body. The main body is where the actual text of the document is created through a combination of normal text and markup. It always begins with the *\begin{document}* command and ends with the *\end{document}* command. Every LaTeX document requires all three of the above command lines irregardless of the contents. Thus, the general structure of a LaTeX document is:

```
\documentclass[options]{class}
    ...

\begin{document}
    ...

\end{document}
```

preamble

body

Ex.)

```
\documentclass{minimal}
\begin{document}
Hello world!
\end{document}
```

3. LaTeX-Syntax

In LaTeX special commands are used for typesetting a text or document. These commands are usually a combination of special characters and letters, do not allow

for any spaces and are case sensitive [1] . Similarly to the valency of a verb in language, LaTeX commands can be classified by the number of arguments they take. In general, they can be categorised into two major groups:

1.) Zero-Argument-Commands
2.) Non-Zero-Argument-Commands

3.1 Zero-Argument-Commands

"Zero-Argument-Commands" are commands that do not require any additional arguments - at least on the surface level - and that can be used intransitively (i.e. on their own). These commands generally consist of three components: 1.) a backslash, 2.) a simple word or phrase indicating the function of the command and 3.) (empty) curly brackets. They can be notated as:

\somecommand{}

Ex.)

\LaTeX{}	–	displays the "LaTeX" symbol (LaTeX)
\noindent{}	–	suppresses paragraph indentation
\bigskip{}	–	creates a vertical empty space

Since basic commands like this do not take on any specified arguments, the curly brackets (for more detail see 3.2) may be left out. While this normally does not entail any loss of function, it can lead to minor behavioural differences (such as inserting or deleting a space), especially when the command is a direct part of the text:

\LaTeX is cool	vs.	*\LaTeX{} is cool*
(= LaTeXis cool)		(= LaTeX is cool)

3.2 Commands and Arguments

"Non-Zero-Argument-Commands" are commands that typically require an argument, i.e. an object they can be applied to. For example, the *\textit{x}* command that is used for italicising needs an object x (usually a word or text) it can refer to or else it would not execute. This type of commands follows a similar syntax to the one seen in 3.1, with the exception that a.) curly brackets cannot be empty and actively require an argument and b.) facultative arguments (arguments that modify a command but are not required for the command to work) may be passed on via square brackets. Sometimes a command may also require more than one obligatory argument, which can be marked with additional curly brackets:

[1] Case sensitive means that upper- and lowercase are treated as distinct. For example, *command1* and *Command1* would be two different commands and not interchangeable. Case sensitivity is also a common source of error, as commands must be entered as intended (e.g. *\LaTeX{}* vs. ~~*\latex{}*~~).

\somecommand[optional]{obligatory}{obligatory}

Ex. 2)

\textit{Hello world!} – puts specified text into italics
\documentclass[paper=a4]{scrartcl} – sets class of document (and specifies paper size)
\textcolor{red}{Hello world!} – applies specified colour to object

3.3 Line Breaks, Spaces and Indents

LaTeX mainly disregards any in-code spaces or line breaks, i.e. they have no visual impact on the actual text document. For example, a sentence can be spread across several lines of code without disrupting the text output. In the same way, any number of spaces can be inserted between code elements without them being transferred to the text document. The only time where spacing matters is for commands, as they cannot be broken up. Compare:

Code	Output
I'm just a simple example trying to make my way through the universe.	I'm just a simple example trying to make my way through the universe.

To specifically include line breaks in a text, one may choose from various options such as:

- <empty line> (creates new line)
- *\newline* (same as <empty line>)
- *\hfill \break* (same as <empty line>)
- *\bigskip* (creates vertical space, ordinarily one line)

Code	Output
You don't need to see my identification. \newline I'm the example you're looking for. \hfill \break I can go about my business. \bigskip{} Move along!	You don't need to see my identification. I'm the example you're looking for. I can go about my business. Move along!

For horizontal spacing some of the more widely spread commands include:

- <empty space> (inserts a standard horizontal space)
- *\enspace* (inserts a horizontal space of 0.5em)
- *\quad* (inserts a horizontal space of 1em; can be stacked)
- *\qquad* (inserts a horizontal space of 2em; same as *\quad \quad*)
- *\hskip<len>* (inserts a space of specified length; can be negative)
- *\hspace{<len>}* (inserts a space of specified length + 1 standard space; can be negative)
- *\hfill* (inserts rubber space that stretches acc. to available space)

Code

```
Hello there!

Hello \enspace there!

Hello \quad there!

Hello \qquad there!

Hello \hskip1.2cm there!

Hello \hspace{1.2cm} there!

\hspace{0.9cm} Hello \hspace{-2.1cm} there!

You \hfill are \hfill a \hfill bold \hfill one.
```

Output

Hello there!
Hello there!
Hello there!
Hello there!
Hello there!
Hello there!
there! Hello
You are a bold one.

Outside of environments (see XXX) the first line of a paragraph is always indented by default. To manually insert indents one may use horizontal spacing commands such as the above, while the *\noindent* command removes the default indentation:

Code

```
Strong I am with indenting, but not that strong. Twilight is upon me and soon night must fall. \bigskip

\noindent Soon I will rest. Yes, forever

\hspace{0.3cm} sleep. Earned it, I have.
```

Output

Strong I am with indenting, but not that strong. Twilight is upon me and soon night must fall.

Soon I will rest. Yes, forever sleep. Earned it, I have.

Finally, in-code indents may serve as a simple way to structure and organise the code to make it more readable and accessible.

3.4 Comments, Special Characters and Quotation

Comments in programming are explanations or annotations in the source code that intend to make the code easier for human readers to understand and that are usually ignored by the compiler or interpreter (and hence not executed). In LaTeX comments are marked through the percentage sign (%).

Code

```
% Sets text to italics

\textit{Comment or comment not. There is no try.}
```

Output

Comment or comment not. There is no try.

Special characters like \, $ or % are often used as operators in LaTeX and thus need a little bit of work around to be included in-text. Some characters can be used by simply prefixing a backslash, others require a unique command:

Special Characters			
Backslash Commands		**Unique Commands**	
Command	**Character**	**Command**	**Character**
\#	#	\textbackslash	\
\$	$	\textasciicircum	^
\%	%	\textasciitilde	~
\&	&	\infty	∞
\{	{	\triangle	△
\}	}	...	...
_	_		
...	...		

Quotation marks are prone to displaying errors, depending on the editor that is used. One of the safest ways to include quotation marks in a text is via the *\enquote{}* function from the 'csquotes' package, which can be conjoined with the *\foreignlanguage{}* function from the 'babel' package to allow for language appropriate quotation. Both packages need to be loaded in via the *\usepackage{}* command (see XXX) in the preamble and can be modified with optional arguments such as the desired languages for the 'babel' package and different styles for the 'csquotes' package (for example, 'autostyle' continuously adapts the quote style to the current document language).

The standard command *\enquote{}* inserts double quotation marks, but can be modified with an asterisk to switch to single quotation marks (i.e. *\enquote*{}*). Finally, quotes may be nested with *\enquote{\enquote{}}* to achieve quotation within quotation.

Code

```
% Load packages in preamble

\usepackage[english, french, german]{babel}
\usepackage[autostyle]{csquotes}

\foreignlanguage{english}{\enquote{Do or do not. There is no try.}} \bigskip

\foreignlanguage{english}{\enquote*{Do or do not. There is no try.}} \bigskip

\foreignlanguage{german}{\enquote{Do or do not. There is no try.}}

\enquote{Yoda, \enquote{Darth Vader} and Luke}
```

Output

"Do or do not. There is no try."

'Do or do not. There is no try.'

„Tu es oder tu es nicht. Es gibt kein Versuchen."

"Yoda, 'Darth Vader' and Luke"

3.5 Environments

In LaTeX environments are macros that can be used to apply specific formats to a section of the document, for example to align or centre text, to generate a table or to

create lists. To initiate an environment the *\begin{}* command is used, which must be closed with the *\end{}* command. Options must be placed after the environment name to avoid any collisions. Finally, environments can in themselves contain (further) environments, which, however, must be closed before the overarching environment ends:

```
\begin{environment1}[options]
    \begin{environment2}[options]
        ...
    \end{environment2}
\end{environment1}
```

Code	Output

`\begin{center}`
 	
You've failed, your highness.
I'm an example,
like my father before me.
 	
`\end{center}`
	You've failed, your highness.
	I'm an example, like my father
	before me.

Some of the most important environments are:

- Text Alignment
  - ***flushleft*** (left aligns the text)
  - ***center*** (centres the text)
  - ***flushright*** (right aligns the text)
- Enumerations/Lists
  - ***itemize*** (creates a list with bullet points)
  - ***enumerate*** (creates a list with numbers)
  - ***description*** (creates an unmarked list of a bold and normal element)
- Graphics
  - ***figure*** (allows for embedding of graphical elements such as images)
  - ***tabular*** (creates a table)
- Others
  - ***document*** (creates the main body of the document)
  - ***verbatim*** (displays the source text without being interpreted)

3.6 Nesting

Sometimes it is desirable to apply several commands to an object at once (for example, when combining bold font with italics). In LaTeX this can simply be achieved by 'nesting' (or 'stacking') commands:

\command1{\command2{…}}

4. Packages

Packages are macros that supplement the base functionality of LaTeX by adding on new functions or features. To use packages they first need to be loaded in, which can be done by including them in the preamble via the *\usepackage{}* directive. Once a package is loaded in, any function from that package may be called upon (e.g. the *\enquote{}* function from the 'csquotes' package). While the order in which packages are listed in the preamble is generally free, it may matter depending on package dependencies (especially for language-related packages). Below is a list of some common packages:

Useful packages		
Package	**Function**	**Additional Information**
rerunfilecheck	provides additional rerun warnings if auxiliary files have changed	files to be checked can be passed on as optional arguments; *[aux]* turns on checking for all auxiliary files
fontspec	for extensive font settings and customisation	
babel	enables multilanguage support for document and ensures typographical accuracy for a variety of languages (e.g. quotation marks, dates)	languages to be used in the document can be passed on as optional arguments via *[ ]*; the last language listed is the main language of the document
hyperref	useful for creating PDFs and cross-references (e.g. hyperlinks, bookmarks, page numbers)	*[unicode]* may be passed on to keep the pdf string in unicode encoding
amsmath	provides extensive options for displaying and customising mathematical equations	
tikz	for creating simpler and more complex graphical elements (e.g. lines, dots, geometric shapes, paths) in LaTeX	
microtype	enhances general appearance of document (e.g. protrusion, interword spacing, kerning)	

Not only are there countless packages to choose from in LaTeX, but individual packages can get quite extensive on their own, as they usually offer a great array of different options and settings. It is nearly impossible to keep track of all the different . Luckily, most LaTeX packages are well-documented and offer a source.

texdoc <package>