LaTeX Typesetting

Björn Lindenberg
Jonas Nordqvist

November 12, 2018

Contents

Introduction

Introduction

LaTeX is a programming language with the following properties:

- It is a document preparation system, and serves as a tool for typesetting journal articles, technical reports, books, and slide presentations.
- It is most suitable for scientific writing, with advanced typesetting of mathematics.
- It is free and available for most common operating systems.

Note that LaTeX is different from word processors, what you see is NOT what you get. Instead it encourages high-quality content in your document, since the focus is not on cosmetics.

Why use LaTeX?

- It “enforces” proper and better typesetting, especially for inexperienced writers.
- Multi-lingual typesetting.
- Automatic generation of bibliographies and indexes.
- Easy to handle large structured documents (*chapters, sections, subsections, index, appendix, table of contents, etc.*)

If you don’t want to think about formatting and want the software to make things look good for you, use LaTeX.

Software

Software

A small selection of distributions:

- TEX Live (Linux, MacOSX, Unix and Windows)
- MiKTEX (Windows)
- MacTEX (MacOSX)
- TEXnicCenter
- TEXmaker or TEXstudio
- shareLATEX(Online)

Literature

Literature

Books

- Per Jacobsson, *Introduktion till LaTeX*, Studentlitteratur, 2004.
- Leslie Lamport, *LaTeX. A Document Preparation System*, Addision-Wesley, 1994.
- Helmut Kopka and Patrick W. Daly, *Guide to LaTeX 2$_\varepsilon$*, Pearson, 2003.
- Frank Mittlebach et al, *The LaTeX Companion*, Addison-Wesley, 2004.
- Michel Goossens et al, *The LaTeX Graphics Companion*, Addison-Wesley, 1997.
- George Gratzer, *Math Into LaTeX*, Birkhauser, 2000.

Internet

- TEX Users Group
- TEX Frequently Asked Questions
- TEX.Stack Exchange
- The TEX Catalogue Online
- TEXnik
- Ti*k*Z web site.

Manuals

- The Not So Short Introduction to LATEX 2$_\varepsilon$
- LATEX Tutorials – A Primer
- Formatting Information
- LATEX 2$_\varepsilon$ for authors
- An essential guide to LATEX 2$_\varepsilon$ usage
- Mathmode
- The Comprehensive LATEX Symbol List
- Using Imported Graphics in LATEX 2$_\varepsilon$.

Compiling a Classic Example

Compiling a Classic Example

We can save the following code:

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

in a standard text file, with the file name `hello.tex`.

Compiling a Classic Example

To typeset our document we have to compile `hello.tex`:

Emacs	`C-c C-c LaTeX`
Terminal window	`latex hello.tex`
TEXnicCenter	Ctrl+F7

Something like the following is shown on the screen:

```
This is TeX, Version 3.14159 (Web2C 7.3.7x)
(./hello.tex
LaTeX2e <2001/06/01>
Babel <v3.7h> and hyphenation patterns for english, dumylang,
nohyphenation, ukenglish, swedish, loaded.
(/usr/share/TeX/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX document class
(/usr/share/TeX/texmf/tex/latex/base/size10.clo)) (./hello.aux) [1]
(./hello.aux) )
Output written on hello.dvi (1 page, 232 bytes).
Transcript written on hello.log.
```

We receive the files `hello.aux`, `hello.dvi` and `hello.log`.

Compiling a Classic Example

- Data for creating cross references is stored in `hello.aux` and in `hello.log` the compilation text is stored.
- The typeset version of `hello.tex` is written to `hello.dvi`.
- You generate a PDF file using either

  ```
  pdflatex hello.tex
  ```

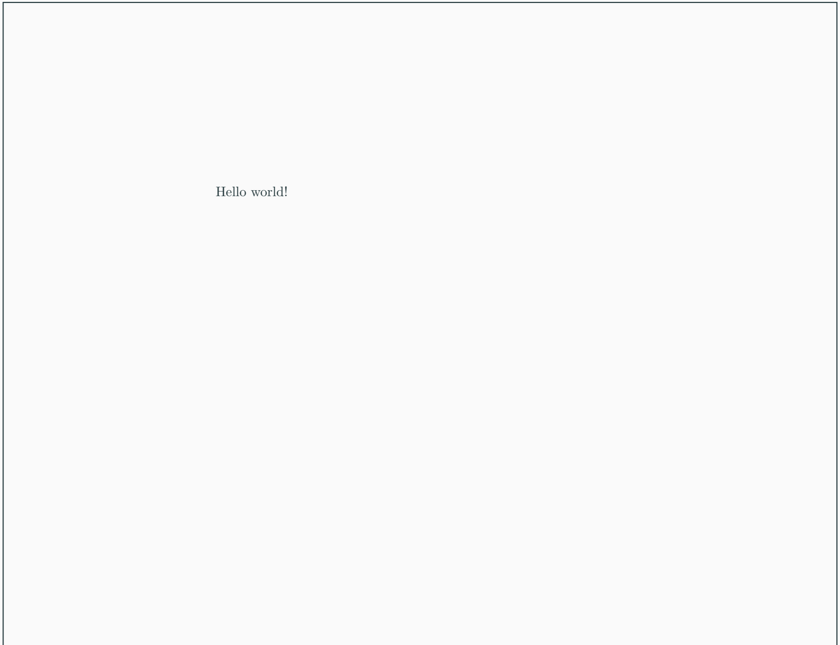
  or

  ```
  ps2pdf hello.ps
  ```

- To create a PostScript file you execute

  ```
  dvips hello.dvi
  ```

Compiling a Classic Example



Hello world!

Syntax

Syntax

Special signs

You control LaTeX using commands beginning with the special sign backslash (\) followed by one or more letters.

Example: `\LaTeX` $\rightarrow$ LaTeX .

Note that LaTeX is case sensitive in command names, e.g., `\LaTeX` and `\latex` are two separate commands.

A command can also begin with backslash followed by a character that is not a letter, e.g., `\,` and `\␣`.

Several space characters following each other is interpreted as one and if it comes after a command, the command “eats” the space.

Example: `a  b   c` $\rightarrow$ a b c

Example: `\LaTeX is fun!` $\rightarrow$ LaTeXis fun!

Use `\␣` to insert a space character after a command.

Example: `\LaTeX\ is fun!` $\rightarrow$ LaTeX is fun!

Curly brackets (`{}`) are used for grouping, e.g., around command arguments.

Example: `\textbf{bold text}` $\rightarrow$ **bold text**

Groups can be used to make local changes.

Example: `abc {\tiny abc} abc` $\rightarrow$ abc abc abc

White space after command arguments does not disappear.

Example: `\textit{italic} style` $\rightarrow$ *italic* style

Special characters like å, ä and ö are not allowed in commands.

All special characters

	Explanation	Command in LaTeX
`\`	backslash	`\textbackslash`
`{`	left brace	`\{`
`}`	right brace	`\}`
`%`	percent character	`\%`
`~`	tilde	`\textasciitilde`
`$`	dollar character	`\$`
`_`	under score	`\_`
`^`	circumflex	`\textasciicircum`
`&`	et-character	`\&`
`#`	number sign	`\#`

All text on a row after a percent sign is ignored by LaTeX, i.e. a comment.

Tilde means space where line break is not allowed.

With \\ you insert a line break.

Environments

For more complicated objects, such as lists and tables, environments are used:

```
\begin{name}
  Text or commands.
\end{name}
```

Where *name* specifies the used environment, e.g., `center`.

Nested environments are possible, make sure they are closed in the correct order. The following is **not** correct:

```
\begin{environment1}
 \begin{environment2}
  This won't work!
  \end{environment1}
\end{environment2}
```

Classes and Packages

Classes and Packages

Structure of a LaTeX file:

```
\documentclass[options]{class}
declarations or preamble
\begin{document}
contents
\end{document}
```

Using *class* you specify the type of document.

The standard classes are `article`, `book`, `report` and `letter`.

Use *options* to choose, for instance, paper format and font size.

Example: A double-paged book in A5 format, with two columns with body text at 11 points font size:

```
\documentclass[twoside,a5paper,twocolumn,11pt]{book}
```

Observe that [*options*] may be left out.

All class options for the standard classes

10pt, 11pt, 12pt	font size of body text
letterpaper, legalpaper, executivepaper, a4paper, a5paper, b5paper	paper format
landscape	landscape paper orientation
onecolumn, twocolumn	number of columns
oneside, twoside	single or double paged
openright, openany	which pages a chapter may start on
notitlepage, titlepage	title etc. on separate page
leqno	formula numbering to the left instead of to the right
fleqn	separate formulas justified to the left, instead of centered
openbib	more spacious formatted bibliography
draft, final	draft or final document version

In the *declaration part* you load packages to get extra functionality. Here you also can define your own commands.

You use the command `\usepackage` to load packages.

Example: Swedish documents should begin[1] :

```
\usepackage[T1]{fontenc}
\usepackage[swedish]{babel}
\usepackage[utf8]{inputenc}
```

The main content is then placed within the `document` environment:

```
\begin{document}
   Main body of text...
\end{document}
```

[1]For Mac users utf8 and Swedish letters etc. might sometimes generate errors due to the automatic inputencoding of Latin1 in editors

Document Structure

Document Structure

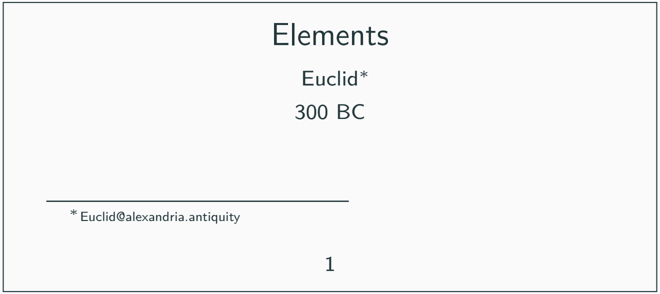
Title and author:

```
\title{Elements}
\author{Euclid\thanks{Euclid@alexandria.antiquity}}
\date{300 BC}
\maketitle
```

The argument to `\thanks` is shown as a footnote. This command may also be used in `\title` and `\date`. You may leave `\date` out, and today's date will be used.

In case of multiple authors `\and` should be used:

```
\title{Music in \LaTeX}
\author{Carl Michael Bellman\\ Sweden \and
  Johann Sebastian Bach\\ Germany}
\maketitle
```

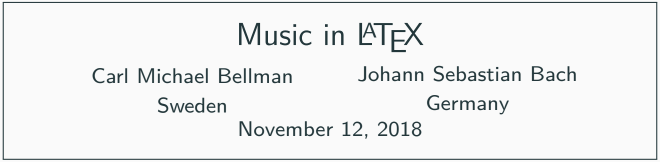

Elements

Euclid*

300 BC

* Euclid@alexandria.antiquity

1


Music in $\LaTeX$

Carl Michael Bellman
Sweden

Johann Sebastian Bach
Germany

November 12, 2018

Abstract:

```
\begin{abstract}
  A short text describing the contents of the document.
\end{abstract}
```

Table of contents:

```
\tableofcontents
```

Other listings: `\listoffigures` and `\listoftables`.

Headings:

```
\part{heading}
\chapter{heading}      \section{heading}
\subsection{heading}   \subsubsection{heading}
\paragraph{heading}    \subparagraph{heading}
```

The command `\chapter` is not defined for the `article` class.

With an asterisk the heading is not numbered, e.g., `\section*{`*heading*`}`.

Example: An article is started with the following heading structure:

```
\section{Divisibility}
\subsection{The Division Algorithm}
\subsection{Prime numbers}
\section{Congruence}
\subsection{The Chinese Remainder Theorem}
```

Which renders the result:

1 Divisibility

1.1 The Division Algorithm

1.2 Prime numbers

2 Congruence

2.1 The Chinese Remainder Theorem

- Unnumbered headings are not automatically included in the table of contents.
- All headings after `\appendix` are numbered as appendices.
- The body text is written after each heading, justified to the right.
- A return character in the text is interpreted as a space character by LaTeX. Page breaks are handled automatically as well.
- One or more empty lines in the body text is interpreted as a new paragraph.
- Every paragraph begins on a new row and is marked with an indent on the first row, except for the first paragraph after a heading.

Please note that no additional space is used to separate two paragraphs, as default.

Typeface

Typeface

Typefaces

Namn	Kommando	Example
Upright	`\textup{`*text*`}`	The quick brown fox
Italic	`\textit{`*text*`}`	*The quick brown fox*
Slanted	`\textsl{`*text*`}`	*The quick brown fox*
Bold	`\textbf{`*text*`}`	**The quick brown fox**
Sanserif	`\textsf{`*text*`}`	The quick brown fox
Small Caps	`\textsc{`*text*`}`	The quick brown fox
Typewriter	`\texttt{`*text*`}`	`The quick brown fox`

These commands can be combined.

Example: `\textit{\textsf{Abcdef}}` → *Abcdef*

The command `\emph{`*text*`}` typesets *text* in italic, if the surrounding environment is not already set to italic, then *text* is set upright.

Example: `\textit{aa \emph{bb} cc}` → *aa* bb *cc*

Sizes

`\tiny`	Example	`\large`	Example
`\scriptsize`	Examplel	`\Large`	Example
`\footnotesize`	Example	`\LARGE`	Example
`\small`	Example	`\huge`	Example
`\normalsize`	Example	`\Huge`	Example

These commands do not take arguments; they are used in groups.

Example: `{\Large 1 {\tiny 2} 3\small 4} 5` $\rightarrow$ 1 2 34 5

Computer Modern is the typeface used as default in LaTeX. But there are other typefaces to choose from. Some of these are:

Name	Package	Example
Bookman	`bookman`	Flygande bäckasiner
Charter	`charter`	Flygande bäckasiner
Concrete	`ccfonts`	Flygande bäckasiner
Helvetica	`helvet`	Flygande bäckasiner
New Century Schoolbook	`newcent`	Flygande bäckasiner
Palatino	`mathpazo`	Flygande bäckasiner
Times	`mathptmx`	Flygande bäckasiner

Do you want to use Palatino you add the row

```
\usepackage{mathpazo}
```

in the declaration part of your document.

Justify Text

Justify Text

To center a text horizontally you write:

```
\begin{center}
  text
\end{center}
```

Note that this environment inserts some space before and after the centered text. You can also use `\centering` inside an environment or a group.

A left justified text, i.e., a text with an uneven right edge, is achieved through:

```
\begin{flushleft}
  text
\end{flushleft}
```

The corresponding command to use inside an environment is `\raggedright`.

With the environments `flushright` or with the command `\raggedleft` a right justified text is achieved.

Example:

```
\begin{flushleft}
  Left
\end{flushleft}
\begin{center}
  Centered
\end{center}
\begin{flushright}
  Right
\end{flushright}
```

Left

Centered

Right

Note that no hyphenation is performed when text is justified as above, which may cause a very uneven edge of the text.

To typeset a longer quotation there are two environments, `quote` and `quotation`, for example:

```
\begin{quote}
  text
\end{quote}
```

These typeset *text* separately with larger margins than the body text.

If *text* consists of multiple paragraphs, these are marked with vertical space or indentation depending on if we use `quote` or `quotation`.

Typesetting poems there is still another environment: `verse`.

In this environment each row is given a specific indentation until the next occurrence of `\\`. Each verse is marked with empty rows.

Lists

Lists

The sections in a list can be numbered or marked with an ornament, e.g., a bullet point.

A numbered list:

1. The first part of this list is a demonstration of how a long row of text is formatted.
2. The second is shorter.
3. The third is almost as short.

The list above was typeset using the code:

```
\begin{enumerate}
  \item The first part ... is formatted.
  \item The second is shorter.
  \item The third is almost as short.
\end{enumerate}
```

Lists with each item marked with an ornament is achieved with the `itemize` environment. Apart from this, the syntax is the same.

An item in a list may contain another list up to four levels.

Example:

```
\begin{itemize}
  \item Section 1.
    \begin{itemize}
      \item Subsection 1.
      \item Subsection 2.
    \end{itemize}
  \item Section 2.
\end{itemize}
```

- Section 1.
  - Subsection 1.
  - Subsection 2.
- Section 2.

In the `description` environment you may select the marking of each item using the command `\item[`*text*`]`.

Example: The code

```
\begin{description}
  \item[InDesign] Layout program.
  \item[\LaTeX] Typesetting program.
  \item[Word] Toy program.
\end{description}
```

gives the result:

InDesign Layout program.

LaTeX Typesetting program.

Word Toy program.

Bits and Pieces

Bits and Pieces

Footnotes are inserted using the command

```
\footnote{text}
```

where the reference sign should be. The *text* appears at the bottom of the page, above the foot margin.

Margin texts are inserted using the command

```
\marginpar{text}
```

where *text* is placed in the left or right margin depending on if we write a single page or double paged document.

Long documents may be split over several smaller files. To insert a *file* you use one of the following commands:

```
\input{file}
\include{file}
```

Hyphenation is done automatically, but sometimes it may be necessary to specify the hyphenation for specific words.

If we write `ny\-qvist\-ism` we will get any of the two hyphenations “ny-qvistism” or “nyqvist-ism”, if needed.

We can also, in the definition part of the document, specify how one or several words should be hyphenated:

```
\hyphenation{ny-qvist-ism ego-tripp-ad}
```

Multiple words are separated using space.

The `babel` package, with `swedish` as argument, provides even more control. If you write `"ff` this is interpreted as `ff-f` at the time of hyphenation. This works for the letters b, d, f, g, l, m, n, p, r, s and t.

Example: The code `stra"ffånge` gives when hyphenated “straff-fånge”, otherwise “straffånge”.

Stresses, letters and characters

ò	`\‘{o}`	ó	`\’{o}`	ô	`\^{o}`	ö	`\"{o}`
õ	`\~{o}`	ō	`\={o}`	ȯ	`\.{o}`	ŏ	`\u{o}`
ǒ	`\v{o}`	ő	`\H{o}`	o͡o	`\t{oo}`	o̧	`\c{o}`
ọ	`\d{o}`	o̲	`\b{o}`	¿	`?‘`	¡	`!‘`
¶	`\P`	§	`\S`	£	`\pounds`	€	`\euro`
œ	`\oe`	Œ	`\OE`	æ	`\ae`	Æ	`\AE`
å	`\aa`	Å	`\AA`	ø	`\o`	Ø	`\O`
ł	`\l`	Ł	`\L`	ß	`\ss`		

Note that `\euro` needs the `eurosym` package.

Hyphen and dash are produced from - respectively `--`. It is also possible to get an extended dash, `---`, rarely used in written Swedish, but often in English.

Example: `IT technology -- a cliché` $\rightarrow$ IT technology – a cliché

Quotations common in Swedish texts:

”	`''`	quotation character
’	`'`	apostrophe (inner quotation)
”’	`''\,'`	apostrophe directly after quotation character
’”	`'\,''`	quotation character directly after apostrophe
«	`<<`	gees eyes, left
»	`>>`	gees eyes, right

Example: according to *Swedish writing rules* quotation are marked like this:

”Tolle lege!”
»Tag och läs!»
»Skynda långsamt.«
” ’Festina lente’ som Augustus sade.”

Note that in English and German text you mark quotations in different ways: “quotation”, using ` `` ` and `''`, and „Anführung“, using `\quotedblbase` and ` `` `, respectively.

Page breaks may be inserted using one of the following commands:

```
\clearpage
\cleardoublepage
\newpage
```

The two first commands place all non-placed floating objects before the page break.

Vertical space are retrieved using three default commands:

```
\bigskip
\medskip
\smallskip
```

or with

\vspace{*length*}

where *length* specifies how much space should be inserted, e.g., `1cm` or `5.3mm`.

Horizontal spaces are retrieved using the command

```
\hspace{length}
```

or with

```
\,
```

Example: `a\,b` → a b

Ellipses or so-called three-points are retrieved using

```
\ldots
```

Example: `a \ldots\  b.` → a … b

Never write three periods since that gives: a ... b.

Today's date is produced by `\today`.

Change Settings

Change Settings

The possibilities to change the ways how things appear is almost unlimited. Here we give a few examples.

To change the settings for the whole document, they should be placed in the declaration part of the document.

Indent is used to mark a new paragraph. If we write

```
\setlength{\parindent}{3em}
```

each indent will be a space as wide as three em-lines in the current typeface and size (an em-line is retrieved from `---`).

Numbering of chapters is controlled by the counter `secnumdepth`. If we set

```
\setcounter{secnumdepth}{1}
```

a heading inserted using `\section` is numbered, but not the headings on lower levels, such as `\subsection`, etc.

Using the counter `tocdepth` we may control which heading levels that should be included in the table of contents.

The representation of the numbering of items in lists of the type `enumerate` are controllable in this way:

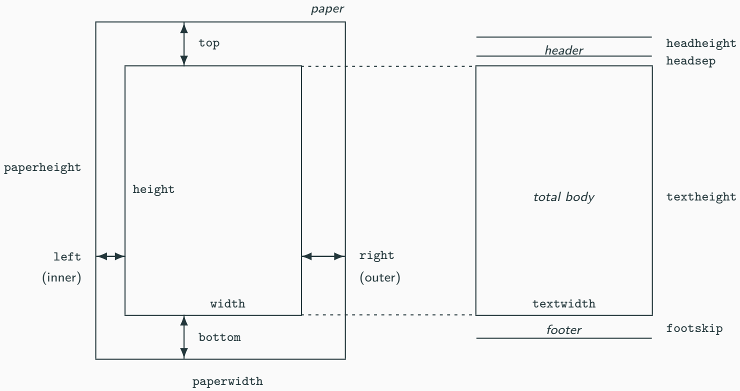
```
\renewcommand{\theenumi}{\Roman{enumi}}
\renewcommand{\theenumii}{\alph{enumii}}
\renewcommand{\theenumiii}{\roman{enumiii}}
\renewcommand{\theenumiv}{\arabic{enumiv}}
```

which gives us the following representations:

- Level 1 – upper case roman letters.
- Level 2 – lower case letters.
- Level 3 – lower case roman numbers.
- Level 4 – numbers.

The margins etc. is easiest defined using the geometry package. We may write like this:

```
\usepackage[a3paper,twoside]{geometry}
\geometry{left=5cm,right=4cm,top=5cm,bottom=6cm}
```



Mathematics

Mathematics

The $\mathcal{A}\!\mathcal{M}\!\mathcal{S}$-LaTeX package is recommended when typesetting mathematics:

```
\usepackage{amsmath,amssymb,amsthm,upref}
```

For further information about typesetting formulas see the manual for $\mathcal{A}\!\mathcal{M}\!\mathcal{S}$-LaTeX.

Mathematic formulae may be inserted using two techniques:

- On a row of text – between dollar characters.
- Floating – in a mathematics environment, such as `equation`.

Many of the commands that are presented in this section only work in a mathematics environment, including between dollar characters.

The two special characters underscore (_) and circumflex (^) work as binary operations, that in relation to the character before it lowers or raises, respectively, the character directly after it.

Example: `$f_n$` $\rightarrow$ f_n

Example: `$x^{10}$` $\rightarrow$ x^{10}

Example: `$e^ni$` $\rightarrow$ e^ni

Example: `$k_{n^2}^{a_j}$` $\rightarrow$ $k_{n^2}^{a_j}$

Example: `list$_n$` $\rightarrow$ list$_n$

Stand-alone formulas are typeset using the environments:

`equation/equation*`	Without line break.
`gather/gather*`	Multiple rows are centered.
`align/align*`	Multiple rows, justified.

There are more environments, but these are the most common.

Note that `\[ ...  \]` is short for the environment `equation*`.

Example: The code

```
\begin{equation}
  \int_{-\infty}^\infty f(x + t) e^{\pi i t} \, dt
\end{equation}
```

gives the result:

$$\int_{-\infty}^\infty f(x + t) e^{\pi i t} \, dt \tag{1}$$

If we replace `\begin{equation}` with `\[` and `\end{equation}` with `\]`, the equation numbering will disappear.

With `gather` each row is centered, and line break is given by `\\`.

Example: This code

```
\begin{gather}
  |x + y| \leq |x| + |y| \\
  f_{n + 2} = f_{n + 1} + f_n \\
  a a^{-1} = e,
\end{gather}
```

gives the result:

$$
\begin{gathered}
|x+y| \leq |x|+|y| \qquad (2) \\
f_{n+2} = f_{n+1} + f_n \qquad (3) \\
aa^{-1} = e, \qquad (4)
\end{gathered}
$$

If we do not wish to have the numbering, we use the `gather*` environment instead.

With `align` we can justify each row along a vertical line.

Example: The code

```
\begin{align*}
  |x + y|   & \leq |x| + |y|    \\
  f_{n + 2} & = f_{n + 1} + f_n \\
  a a^{-1}  & = e,
\end{align*}
```

gives the result:

$$\begin{aligned}
|x + y| & \leq |x| + |y| \\
f_{n + 2} & = f_{n + 1} + f_n \\
a a^{-1} & = e,
\end{aligned}$$

Note how `&` is used. This environment is suitable when you want to show multiple steps of a long calculation. If numbering is wanted, you simple remove the asterisk (`*`).

Rational expressions are typeset using the command

```
\frac{numerator}{denominator}
```

where *numerator* and *denominator* may be arbitrary expressions. To get a binomial coefficient you exchange `\frac` for `\binom`.

Example: The code

```
\[
  \binom{n}{k} = \frac{n!}{k! (n - k)!}.
\]
```

gives the result:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Note that a space character in mathematics mode does not affect the final result and may thus be used to enhance the readability of the code.

The root character is retrieved by the command:

\sqrt{*expression*}

Example: `$\sqrt{2x}$` $\rightarrow$ $\sqrt{2x}$

Alternatives are typeset using the environment `cases`.

Example:

```
|x| = \begin{cases}
        -x & \text{if $x < 0$} \\
         x & \text{else.}
      \end{cases}
```

where `\text` temporarily exits mathematics mode. The code above gives the result:

$$|x| = \begin{cases} -x & \text{if } x < 0 \\ x & \text{else.} \end{cases}$$

In the code above, the actual code for the mathematics environment is left out, and this will be the case for the following examples, with very special exceptions.

Matrices consist of rows and columns. For row and column separators `\\` and `&` are used, respectively.

Example: Matrices

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}, \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix}, \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix}, \begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix}$$

are typeset using the code

```
\begin{matrixType}
  1 & 2 \\
  3 & 4
\end{matrixType}
```

where *matrixType* in order of appearance is `matrix`, `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` and `Vmatrix`.

Parentheses exists of various types. Below follow the most common:

$()$	`()`	$\{\}$	`\{\}`
$\|\|$	`||`	$[]$	`[]`
$\\|\\|$	`||`	$\langle\rangle$	`\langle\rangle`
$\lceil\rceil$	`\lceil\rceil`	$\lfloor\rfloor$	`\lfloor\rfloor`

For the parentheses to be of the right size you use `\left` and `\right` before left and right parenthesis, respectively.

Example: The code

```
(\frac{1}{n + 1} - k^2)^{1/2}
\left(\frac{1}{n + 1} - k^2\right)^{1/2}
```

gives the result:

$$(\frac{1}{n+1} - k^2)^{1/2} \quad \text{respectively} \quad \left(\frac{1}{n+1} - k^2\right)^{1/2}.$$

Observe that `\left` and `\right` must come in pairs of two. With a period character you insert an “open” parenthesis.

Example: From

```
\left\lceil\frac{1}{x + 1} - \frac{x}{x^2 - 1}\right.
\left\{1 + \frac{1}{p}\right\|
```

we get:

$$\left\lceil\frac{1}{x+1} - \frac{x}{x^2-1}\right. \quad \text{and} \quad \left\{1 + \frac{1}{p}\right\|, \quad \text{respectively.}$$

Congruences are typeset using `\pmod`.

Example: `x^2 \equiv a \pmod{p}` $\rightarrow$ $x^2 \equiv a \pmod{p}$

Alternatives to `\pmod` are `\mod`, `\bmod` and `\pod`.

Spaces

$\rule{1em}{0.3em}\,\rule{1em}{0.3em}$	`\thinspace`	`\,`	$\rule{1em}{0.3em}\!\rule{1em}{0.3em}$	`\negthinspace`	`\!`
$\rule{1em}{0.3em}\:\rule{1em}{0.3em}$	`\medspace`	`\:`	$\rule{1em}{0.3em}\negmedspace\rule{1em}{0.3em}$	`\negmedspace`	
$\rule{1em}{0.3em}\;\rule{1em}{0.3em}$	`\thickspace`	`\;`	$\rule{1em}{0.3em}\negthickspace\rule{1em}{0.3em}$	`\negthickspace`	
$\rule{1em}{0.3em}\quad\rule{1em}{0.3em}$	`\quad`				
$\rule{1em}{0.3em}\qquad\rule{1em}{0.3em}$	`\qquad`				

Ellipses

$\ldots$	`\ldots`	$\cdots$	`\cdots`	$\ddots$	`\ddots`	$\vdots$	`\vdots`

Typefaces

Abc	`\mathrm{`*text*`}`	Abc	`\mathit{`*text*`}`
$\mathbf{Abc}$	`\mathbf{`*text*`}`	Abc	`\mathsf{`*text*`}`
$\mathtt{Abc}$	`\mathtt{`*text*`}`	$\mathcal{ABC}$	`\mathcal{`*text*`}`
$\mathfrak{Abc}$	`\mathfrak{`*text*`}`	$\mathbb{ABC}$	`\mathbb{`*text*`}`
$\boldsymbol{\alpha\beta\gamma}$	`\boldsymbol{`*text*`}`		

Operators and functions

$\arccos$	`\arccos`	$\arcsin$	`\arcsin`	$\arctan$	`\arctan`
$\arg$	`\arg`	$\cos$	`\cos`	$\cosh$	`\cosh`
$\cot$	`\cot`	$\coth$	`\coth`	$\csc$	`\csc`
$\deg$	`\deg`	$\det$	`\det`	$\dim$	`\dim`
$\exp$	`\exp`	$\gcd$	`\gcd`	$\hom$	`\hom`
$\inf$	`\inf`	$\ker$	`\ker`	$\lim$	`\lim`
$\liminf$	`\liminf`	$\limsup$	`\limsup`	$\ln$	`\ln`
$\log$	`\log`	$\max$	`\max`	$\min$	`\min`
$\sec$	`\sec`	$\sin$	`\sin`	$\sinh$	`\sinh`
$\sup$	`\sup`	$\tan$	`\tan`	$\tanh$	`\tanh`

You may define you own operators using the command:

`\DeclareMathOperator*{`*function*`}{`*text*`}`

which is placed in the declaration part of the document.

Example: The code

```
\DeclareMathOperator{\Gal}{Gal}
\DeclareMathOperator*{\minmax}{min\,max}
...
\Gal(L/K),\; \Gal_n(L/K)
\quad\text{and}\quad
\minmax_{x \in A} f(x), \text{respectively}.
```

gives the result:

$$\operatorname{Gal}(L/K),\; \operatorname{Gal}_n(L/K) \quad\text{respectively}\quad \operatorname*{min\,max}_{x \in A} f(x).$$

Note that we may control, using the asterisk, if a superscript or a subscript should be placed above or below the operator.

Accents

$\acute{x}$	`\acute{x}`	$\bar{x}$	`\bar{x}`	$\breve{x}$	`\breve{x}`
$\check{x}$	`\check{x}`	$\ddddot{x}$	`\ddddot{x}`	$\dddot{x}$	`\dddot{x}`
$\ddot{x}$	`\ddot{x}`	$\dot{x}$	`\dot{x}`	$\grave{x}$	`\grave{x}`
$\hat{x}$	`\hat{x}`	$\mathring{x}$	`\mathring{x}`	$\tilde{x}$	`\tilde{x}`
$\vec{x}$	`\vec{x}`	x'	`x'`	x''	`x''`

Accents over multiple characters

`\whidehat`	`\widetilde`
`\overline`	`\underline`
`\overbrace`	`\underbrace`
`\overleftarrow`	`\overrightarrow`
`\underleftarrow`	`\underrightarrow`
`\overleftrightarrow`	`\underleftrightarrow`

Example: `\overrightarrow{AB}` $\rightarrow$ $\overrightarrow{AB}$

Greek letters

α	`\alpha`	β	`\beta`	γ	`\gamma`
Γ	`\Gamma`	δ	`\delta`	Δ	`\Delta`
ϵ	`\epsilon`	ε	`\varepsilon`	$\digamma$	`\digamma`
ζ	`\zeta`	η	`\eta`	θ	`\theta`
ϑ	`\vartheta`	Θ	`\Theta`	ι	`\iota`
κ	`\kappa`	$\varkappa$	`\varkappa`	λ	`\lambda`
Λ	`\Lambda`	μ	`\mu`	ν	`\nu`
ξ	`\xi`	Ξ	`\Xi`	π	`\pi`
ϖ	`\varpi`	Π	`\Pi`	ρ	`\rho`
ϱ	`\varrho`	σ	`\sigma`	ς	`\varsigma`
Σ	`\Sigma`	τ	`\tau`	υ	`\upsilon`
Υ	`\Upsilon`	ϕ	`\phi`	φ	`\varphi`
Φ	`\Phi`	χ	`\chi`	ψ	`\psi`
Ψ	`\Psi`	ω	`\omega`	Ω	`\Omega`

Functions are typeset using `\colon`.

Example: `f \colon A \to B` $\rightarrow$ $f\colon A \to B$

A few common characters

$+$	`+`	$-$	`-`	$\aleph$	`\aleph`
$\angle$	`\angle`	$\bot$	`\bot`	$\cap$	`\cap`
$\cdot$	`\cdot`	$\circ$	`\circ`	$\complement$	`\complement`
$\cup$	`\cup`	ℓ	`\ell`	$\emptyset$	`\emptyset`
$\exists$	`\exists`	$\forall$	`\forall`	$\Im$	`\Im`
∞	`\infty`	$\land$	`\land`	$\lor$	`\lor`
$\mp$	`\mp`	∇	`\nabla`	$\neg$	`\neg`
$\odot$	`\odot`	$\ominus$	`\ominus`	$\oplus$	`\oplus`
$\otimes$	`\otimes`	∂	`\partial`	$\pm$	`\pm`
$\Re$	`\Re`	$\setminus$	`\setminus`	$\times$	`\times`
$\triangle$	`\triangle`	$\wp$	`\wp`	$\wr$	`\wr`

Some relations

$<$	`<`	$>$	`>`	$=$	`=`
$\approx$	`\approx`	$\backsim$	`\backsim`	$\cong$	`\cong`
$\equiv$	`\equiv`	$\geq$	`\geq`	$\geqslant$	`\geqslant`
$\gg$	`\gg`	$\in$	`\in`	$\leq$	`\leq`
$\leqslant$	`\leqslant`	$\ll$	`\ll`	$\neq$	`\neq`
$\owns$	`\owns`	$\prec$	`\prec`	$\sim$	`\sim`
$\simeq$	`\simeq`	$\subset$	`\subset`	$\subseteq$	`\subseteq`
$\succ$	`\succ`	$\supset$	`\supset`	$\supseteq$	`\supseteq`

To negate a relation put the command `\not` before.

Example: `A \not\subset B` $\rightarrow$ $A \not\subset B$

There is another command to denote that something does not belong to something else: `\notin`

The symbol of divisibility is given by `\mid`.

Example: `a \mid b` $\rightarrow$ $a \mid b$

Some arrows

$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`
$\leftarrow$	`\leftarrow`, `\gets`	$\Leftarrow$	`\Leftarrow`
$\leftrightarrow$	`\leftrightarrow`	$\Leftrightarrow$	`\Leftrightarrow`
$\rightarrow$	`\rightarrow`, `\to`	$\Rightarrow$	`\Rightarrow`
$\mapsto$	`\maptso`	$\dashrightarrow$	`\dasharrow`

Some sigma operators

$\displaystyle\int$	`\int`	$\displaystyle\oint$	`\oint`	$\displaystyle\bigcap$	`\bigcap`
$\displaystyle\bigcup$	`\bigcup`	$\displaystyle\prod$	`\prod`	$\displaystyle\sum$	`\sum`

Double and tripple integrals are given by `\iint` and `\iiint`, respectively.

Theorems and such are typeset using environments, they are created by one of the following commands:

\newtheorem{*name*}[*counter*]{*heading*}

\newtheorem{*name*}{*heading*}[*counter*]

Example: With the code:

```
\newtheorem{theorem}{Theorem}[section]
\newtheorem{lemma}[theorem]{Lemma}
```

two environments are created, `theorem` and `lemma`. They are set using the headings "Theorem" and "Lemma", respectively. Both headings are numbered, the first regarding section and the second uses the same counter, e.g., in section 2 we may have the following headings: Theorem 2.1, Lemma 2.2, Lemma 2.3, Theorem 2.4 etc.

By using

```
\theoremstyle{style}
```

before `\newtheorem`, you may control the style of the theorem you create. There are three predefined styles: `plain`, `definition` and `remark`. You may create new styles using `\newtheoremstyle`.

A “theorem” is typeset as an environment:

```
\begin{type}[info]
  text
\end{type}
```

where [*info*] may be left out.

Example: The code

```
\begin{theorem}[Fermat’s last theorem]
  Let $n$ be a natural number. If $n > 2$, there are no
  natural number solutions to $x^n + y^n = z^n$.
\end{theorem}
```

results in:

> **Theorem 1.2 (Fermat’s last theorem).** *Let n be a natural number. If $n > 2$, there are no natural number solutions to $x^n + y^n = z^n$.*

Proofs are typeset using the environment `proof`:

```
\begin{proof}[optional heading]
  text
\end{proof}
```

If [*optional heading*] is left out, the predefined heading "Proof" is used. After *text* a Halmos symbol is inserted, i.e., □.

Example: The code

```
\begin{proof}
  Suppose that $x^n + y^n = z^n$ for some ...
\end{proof}
```

give the result:

> *Proof.* Suppose that $x^n + y^n = z^n$ for some x, y and z. Then the following is true ... and thus the theorem is true.
> □

Tables

Tables

A table is typeset using the environment:

```
\begin{tabular}{columns}
  rows
\end{tabular}
```

where `\\` and `&` are used as row and column separators, respectively. The number of columns and their types are defined by *columns*.

The following column types are possible:

`l`	Aligned to the left.
`c`	Centered.
`r`	Aligned to the right.
`p{`*width*`}`	Column with fixed *width*, e.g., `p{3cm}`. Rows exceeding this width are broken automatically, without affecting the other cells on that row.

Continued on the next slide.

`|`	Vertical line.
`||`	Double vertical line.
`*{`n`}{`col`}`	To insert the same column type, *col*, n times, e.g., `*{4}{c|}` is the same as `c|c|c|c|`.
`@{`$text$`}`	Inserts *text* on each row, between the two columns it appears in *columns*. Observe that the space that is normally inserted between two columns, disappears when you use an `@`-expression.

Apart from these, there is also the command `\vline` which inserts a vertical line on the row it is used.

Horizontal lines are produced using the following command:

```
\hline
\cline{m-n}
```

where `\cline` creates a line from the mth column to the nth.

Example: The table

AAA	BBB	CCC
ddd-ddd	012	x
eee-eee-eee	34567	yyy

was typeset using the following code:

```
\begin{tabular}{|lr|c|}
  \hline
  AAA         & BBB   & CCC \\
  \hline
  ddd-ddd     & 012   & x   \\
  eee-eee-eee & 34567 & yyy \\
  \hline
\end{tabular}
```

Note how we use the space characters to make the code easier to read.

To let a text span multiple columns we use the command:

`\multicolumn{`n`}{`col`}{`$text$`}`

where n give the number of columns that *text* should stretch over and using *col* you control the types of new columns. Allowed values in *col* is exactly one of `l`, `c` and `r`, together with `|` and `@`. It is alright to use `1` as the value of n, which is appropriate when you want to change the alignment of a specific cell.

Example: Suppose we want to write the text "Options" centered in a column that stretches over four columns. Thus we write:

```
\multicolumn{4}{c}{Options}
```

A possible vertical line to the right of the four original columns disappears. If you wish that the line remains, you replace `{c}` with `{c|}`.

Example: Study the following table:

AAAA		
BBBB	CCCCCC	DDD
aaa	Quousque tandem abutere, Catilina, patientia nostra?	0.11:22
bbb	eeee fff gg hhhh	3.44
ccc	iii	55.66
ddd	Flygande bäckasiner söka hvila på mjuka tufwor.	77.88:99

Note the following: (1) The horizontal lines stand out to the right, but not to the left – column type `@{}l`. (2) The second column manages line breaks – column type `p{5.5cm}`. (3) The times are justified according to the dot – column type `r@{.}l`.

The code to the table above:

```
\begin{tabular}{@{}l|p{5.5cm}|r@{.}l}
  \hline
  \textbf{AAAA} \\
  \cline{1-2}
  BBBB  & CCCCCC            & \multicolumn{2}{c}{DDD} \\
  \hline
  \hline
  aaa   & Quousque tandem abutere, Catilina,
          patientia nostra?                  &  0&11:22 \\
  bbb   & eeee fff gg hhhh                   &  3&44    \\
  ccc   & iii                                & 55&66    \\
  ddd   & Flygande b\"ackasiner s\"oka hvila p\aa
          mjuka tufwor.                      & 77&88:99 \\
  \hline
\end{tabular}
```

Example: Sometimes it may be difficult to find the number of columns and which type each should have. Study the following:

Movies

			Grade				
Title	*Year*	*Country*	1	2	3	4	5
Persona	1966	Sweden					✔
Play Time	1967	France					✔
Solaris	1972	USSR					✔
Sleeper	1973	USA				✔	
Pretty Woman	1990	USA	✔				
Matrix	1999	USA		✔			
The Road Home	1999	China				✔	
Lagaan	2001	India				✔	

At most we have eight columns: `|lcl|ccccc|`, see for instance the third row.

```
\begin{tabular}{|lcl|ccccc|}
  \hline
  \multicolumn{8}{c}{\textbf{Movies}}
  \hline
  \hline
  & & & \multicolumn{5}{c|}{\textsf{Grade}}
  \cline{4-8}
  \emph{Title} & \emph{Year}           & \emph{Country} & 1 & 2 & 3 &
  \hline
  Persona       & 1966 & Sweden      & & & & & \ding{52} \\
  Play Time     & 1967 & France      & & & & & \ding{52} \\
  Solaris       & 1972 & USSR        & & & & & \ding{52} \\
  Sleeper       & 1973 & USA         & & & & \ding{52} & \\
  Pretty Woman  & 1990 & USA         & \ding{52} & & & & \\
  Matrix        & 1999 & USA         & & \ding{52} & & & \\
  The Road Home & 1999 & China       & & & & \ding{52} & \\
  Lagaan        & 2001 & India       & & & & \ding{52} & \\
  \hline
\end{tabular}
```

The command `\ding` requires the `pifont` package.

To get lower case numbers the command `\oldstylenums` is used.

Example: `\oldstylenums{1476}` $\rightarrow$ 1476

A few packages suitable for tables:

`array`	More column types.
`supertabular`	Table spanning multiple pages.
`longtable`	Table spanning multiple pages.
`colortbl`	Columns and row with different background colors.
`booktabs`	More professional tables.

Pictures

Pictures

The following picture formats are allowed, depending on compiler:

LaTeX	EPS, MPS
pdfLaTeX	JPEG, MPS, PDF, PNG

where MPS is a variant of EPS, created by MetaPost.

To be able to insert pictures you have to include the following package:

```
\usepackage[drive]{graphicx}
```

where *drive* is either `dvips` or `pdftex` depending on if we use LaTeX or pdfLaTeX, respectively, though it can usually be omitted.

Images are inserted using the command:

```
\includegraphics[parameters]{image file}
```

Note that if the *image file* is in a subdirectory, then you need to add `\graphicspath{{image directory/}}` to the preamble.

With [$parameters$] you manipulate the inserted image, but the construction is left out. You write *parameters* like a list of elements on the form: *keyword*=⟨*value*⟩.

Some common keywords:

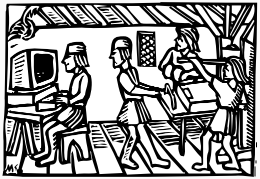
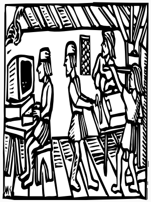
`height`	Forces the height of the image to be ⟨*value*⟩.
`width`	Forces the width of the image to be ⟨*value*⟩.
`scale`	Scales the image using the factor ⟨*value*⟩.
`angle`	Rotates the image ⟨*value*⟩ degrees counter clockwise.
`origin`	The center for the rotation, e.g., ⟨*value*⟩ may be `c` or `rt`.
`viewport`	Defines a window in the image. Here ⟨*value*⟩ must be on the form x_1 y_1 x_2 y_2, where (x_1, y_1) and (x_2, y_2) is the lower left and right corner, respectively.
`clip`	Clips everything outside the window.

For more information see *Using Imported Graphics in LaTeX 2ε*.

Example: We write

```
\includegraphics[width=3.5cm]{typo}
\includegraphics[width=2cm,height=2.75cm]{typo}
```

which yields

 and 

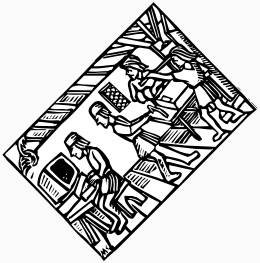
respectively, assuming that we have a file named `typo` and it contains the above image.

Note that if either `height` or `width` is left out, the image is scaled to keep the original aspect ratio intact.

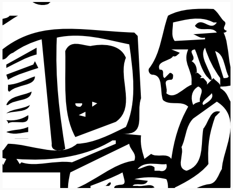
Example: The code

```
\includegraphics[width=3cm,angle=45]{typo}
\includegraphics[viewport=30 145 165 255,
                 clip,width=3cm]{typo}
```

give the result:



respectively



TikZ & PGF

TikZ & PGF

Using the `tikz` package you may create images directly in LaTeX.
You can find the complete manual here:
`http://pgf.sourceforge.net/pgf_CVS.pdf`

Usually you use the following environment:

```
begin{tikzpicture}[scale=2]
  commands for drawing fancy stuff
end{tikzpicture}
```

This creates a drawing area at the current position in the text and scales the content by 2. The scale argument is optional. You can also scale in one dimension or different scaling for two dimensions:

```
\begin{tikzpicture}[xscale=2, yscale=5]
  commands for drawing fancy stuff
\end{tikzpicture}
```

The default drawing unit is 1 cm.

Predefined colors are: red🟥, green🟩, blue🟦, cyan🟦, magenta🟪, yellow🟨, black⬛, gray⬛, darkgray⬛, lightgray⬜, brown🟫, lime🟩, olive🟫, orange🟧, pink⬜, purple🟥, teal🟦, violet🟪, and white⬜.

Example: `{\color{blue} blue text}` $\rightarrow$ blue text

You define your own colors using the commands

```
\newgray{name}{n}
\newrgbcolor{name}{r g b}
```

where $0 \leq n, r, g, b \leq 1$.

Example: Let us define a golden red color:

```
\newrgbcolor{goldred}{0.8 0.61 0.11}
```

and use it:

```
{\goldred golden red text}
```

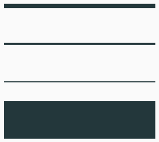
which gives the result: gold red text.

Line thickness, line type, color etc. are controlled using different parameters. Most drawing commands take multiple arguments in square brackets divided by a comma.

For instance, changing the thickness of lines:

```
\begin{tikzpicture}
  \draw [ultra thick] (0,1.5) -- (2,1.5);
  \draw [thick] (0,1) -- (2,1);
  \draw [thin] (0,0.5) -- (2,0.5);
  \draw [line width=0.5cm] (0,0) -- (2,0);
\end{tikzpicture}
```

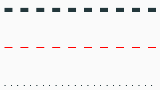
Result:



You can also change the line style:

```
\begin{tikzpicture}
 \draw [dashed, ultra thick] (0,1) -- (2,1);
 \draw [dashed, red] (0, 0.5) -- (2,0.5);
 \draw [dotted] (0,0) -- (2,0);
\end{tikzpicture}
```

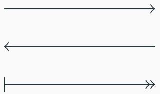
Result:



You can put decorations on lines. For instance you can draw arrows or bars at the start and end of a line:

```
\begin{tikzpicture}
  \draw [->] (0,0) -- (2,0);
  \draw [<-] (0, -0.5) -- (2,-0.5);
  \draw [|->>] (0,-1) -- (2,-1);
\end{tikzpicture}
```

Result:

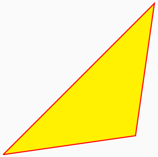


If you do not specify the arrow type - is used. You may combine the different arrow types (see the third line).

To draw a polygon with corners in the points (x_1, y_1), ..., (x_n, y_n):

Example:

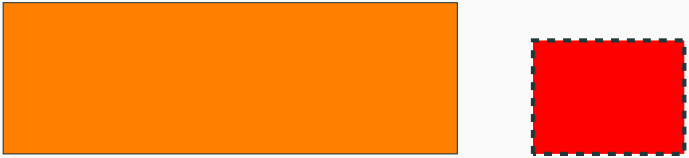
```
\draw [red, fill=yellow] (0,0)--(2,2)--(1.75,0.25)--(0,0);
```



The argument `red` draws a red outline, while `fill=yellow` fills the polygon with specified color.

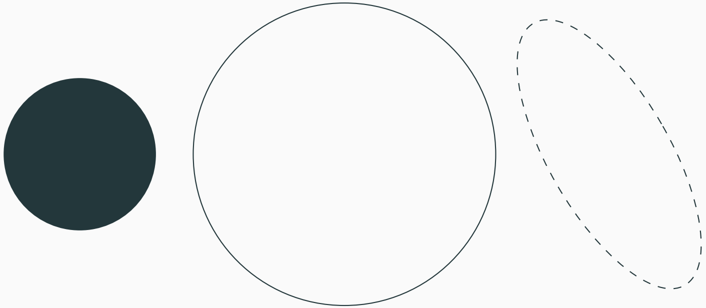
Drawing a rectangle is done with (x_1, y_1) and (x_2, y_2) as opposite corners:

```
\draw [fill=orange] (0,0) rectangle (2,6);
\draw [fill=red, ultra thick, dashed] (7,0) rectangle (9,1.5);
```



Drawing a circle is done by specifying the center (x, y) and the radius r. For an ellipse we specify the x and y radius:

```
\draw [fill] (1,1) circle [radius=1];
\draw (4.5, 1) circle (2);
\draw [dashed] (8, 1) circle
         [x radius=0.8, y radius=2, rotate=30];
```

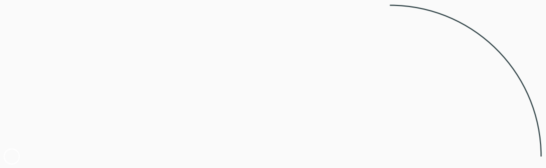


An arc is drawn with the start point at (x, y), a radius r and starts at α_1 degrees and continues counterclockwise until α_2 given in degrees.

\draw (x, y) arc (α_1:α_2:r);

Example:

```
\draw (7,0) arc (0:90:2);
```

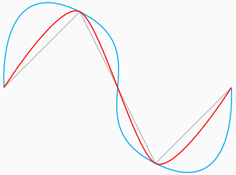


Interpolate an open curve through the points (x_i, y_i):

```
    \draw[param] plot [smooth, tension=2] coordinates (x_1,y_1),
..., (x_i,y_i) };
```

Example:

```
\draw [gray!50]  (0,0) -- (1,1) -- (2,-2) -- (3,0);
\draw [cyan] plot [smooth, tension=2]
      coordinates { (0,0) (1,1) (2,-2)    (3,0)};
\draw [red] plot [smooth, tension=0.5]
      coordinates { (0,0) (1,1) (2,-1) (3,0)};
```

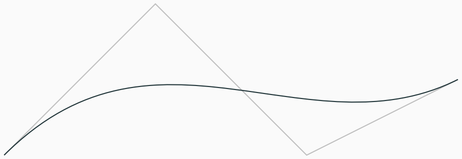


Draw a cubic Bézier curve with (x_1, y_1), ..., (x_4, y_4) as control points:

`\draw[`$param$`](`x_1`,`y_1`)..controls(`x_2`,`y_2`) and (`x_3`,`y_3`)..(`x_4`,`y_4`);`

Example:

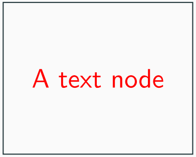
```
\draw [gray!50]  (0,0) -- (2,2) -- (4,0) -- (6,1);
\draw (0,0) .. controls (2,2) and (4,0) .. (6,1);
```



Text labels are done with the **node** object

Example:

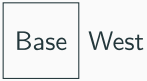
```
\draw (0,0) rectangle (2.5,2);
\node [red] at (1.25,1) {A text node};
```



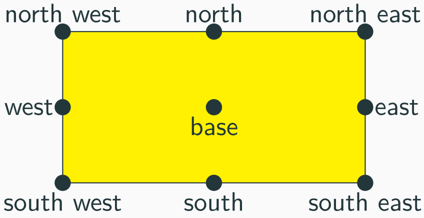
You can also specify the anchor for nodes:

Example:

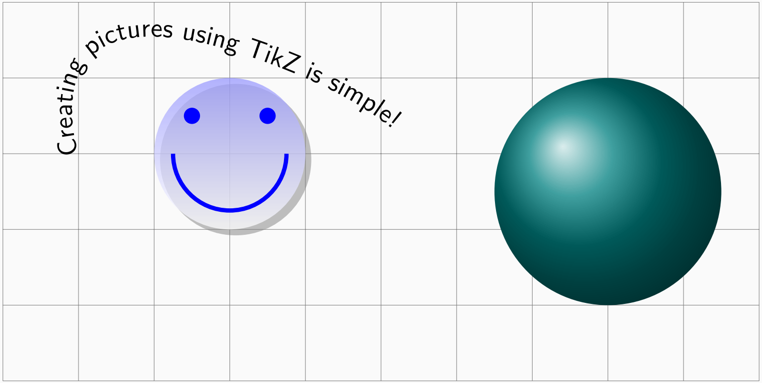
```
\draw (0,0) rectangle (1,1);
\node at(0.5,0.5) {Base};
\node [anchor=west] at (1, 0.5) {West};
```



Predefined anchors are:



A larger example



See next page for the code.

```
\begin{tikzpicture}
  \draw [help lines] (0,0) grid (10,5);
  \draw [drop shadow, decorate,decoration={text along path,
          text={Creating pictures using Tikz is simple!}}]
          plot [smooth, tension=2]
          coordinates { (1,3) (4,4) (8,0) (10,5)};
  \shade[top color=blue!40!white,opacity=0.75, drop shadow]
  (3,3) circle [radius = 1];
  \draw [ultra thick, blue](3.75,3) arc (0:-180:0.75);
  \draw [fill=blue, blue] (3.5, 3.5) circle (0.1);
  \draw [fill=blue, blue] (2.5, 3.5) circle (0.1);
  \shade[ball color=teal] (8,2.5) circle (1.5);
\end{tikzpicture}
```

The drop shadow option requires another package:
`\usetikzlibrary{shadows}`

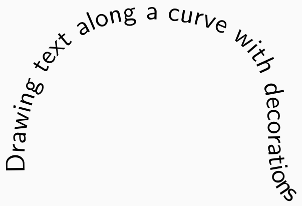
To get text along a curve we need to add another package:
`\usetikzlibrary{decorations.text}`

Then we can add a text decoration along an object by adding it to the arguments of the command:

```
[decorate,decoration={text along path, text={The text}}]
```

Example:

```
\draw [decorate,decoration={text along path,
     text={Drawing text along a curve with decorations}}]
     plot [smooth, tension=2]
     coordinates { (0,0) (2,1) (4,0) (6,2)};
```



Floating Objects

Floating Objects

Figure and tables should be floating, which means that these are placed either at the top or at the bottom of the page.

There are two default environments: `figure` and `table`. This is how they are used:

```
\begin{floating object}[var]
   image or table
\end{floating object}
```

var controls where the object should be placed. Common values are:

`h`	Here – place the object where the environment is placed.
`t`	Top – floating object.
`b`	Bottom – sinking object.
`p`	Page – its own page.

If `[`*var*`]` is left out, the combination `tbp` is used.

Image and table text are inserted using the command

```
\caption[short text]{text}
```

where *short text* is the optional text that is used in a possible listing of images or tables, created with any of the two commands `\listoffigures` and `\listoftables`.

To center the image, the table or associated text horizontally, the command `\centering` should be used.

Example: To make a table floating you write:

```
\begin{table}
  \centering
  \begin{tabular}{columns}
    rows
  \end{tabular}
  \caption{table text}
\end{table}
```

To make an image "floating" is as easy:

```
\begin{figure}[htbp]
  \centering
  \includegraphics[width=3cm]{typo.eps}
  \caption{Gutenberg at work.}
\end{figure}
```

If you use Ti*k*Z you replace `\includegraphics` with the environment `tikzpicture`. The code above gives the result:

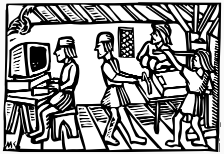


Figure 1. Gutenberg at work.

Note that the heading "Figure" is in English. It would automatically have been "Figur" if we had loaded the package `babel` with `swedish`.

Cross References

Cross References

Multiple objects are automatically numbered in LaTeX, e.g., heading, sections, list items, floating objects, and theorems.

Cross referencing such objects may be handled automatically. Suppose we want to refer to the equation (1.3) and we write the code

```
... according to (1.3) we get that ...
```

But later we realize that we have to add another formula before (1.3), this causes our formula to get a new number. Now, we must change the code – with the risk that we will miss some instance of `(1.3)`.

To be able to refer to an object, we have to tag it with a unique label, using the command:

```
\label{label}
```

Where in the code should `\label` be placed?

Heading	After `\section` etc.
List	After corresponding `\item`.
Formula	Inside the environment. Corresponding row in a multi-row formula.
Theorem	Inside the environment.
Floating object	After or inside `\caption`.

Example:

```
\section{Rings and Bodies}
\label{sec:rings_and_bodies}
\begin{equation}
  \label{eq:abelian}
  a + b = b + a
\end{equation}
```

To refer to a label, we have the following commands:

```
\ref{label}
\pageref{label}
\eqref{label}
```

where the last refers to equations and requires $\mathcal{AMS}$-LaTeX.

Example: Suppose that the heading in the previous example is the second section and that it begins on page 14. Also suppose that the formula is the fourth in this section. We now have:

Code	Result
`in section~\ref{sec:rings_and_bodies}`	in section 2
`on page~\pageref{sec:rings_and_bodies}`	on page 14
`the equation~\eqref{eq:abelian}`	the equation (2.4)

References

References

A bibliography is typeset using the environment:

```
\begin{thebibliography}{example}
  references
\end{thebibliography}
```

which is actually a list, where each reference is an item. The width of *example* gives how much each item should be indented from the left side.

Every reference is started using the command:

```
\bibitem[text]{label}
```

where *text* is used as a mark of the item, and *label* is used to refer to the corresponding reference, cf. `\label`. If `[text]` is left out [1], [2], etc. is used instead.

To refer to a reference in the text, you use the command:

`\cite{`*label*`}`

Example: The result of the code below can be seen on the next page.

```
... Aschbacher~\cite{Aschbacher} ...
Batra and Morton~\cite{BatraMorton} ...

\begin{thebibliography}{9}
\bibitem{Aschbacher} M. Aschbacher, \emph{Finite Group
  Theory}, second edition, Cambridge University Press,
  Cambridge, 2000.
\bibitem{BatraMorton} Anjula Batra and Patrick Morton,
  \emph{Algebraic dynamics of polynomial maps on the
  algebraic closure of a finite field, I}, Rocky
  Mountain J. Math. \textbf{24} (1994), no. 2, 453--481.
\end{thebibliography}
```

. . . Aschbacher [1] . . . Batra and Morton [2] . . .

Bibliography

M. Aschbacher, *Finite Group Theory*, second edition, Cambridge University Press, Cambridge, 2000.

Anjula Batra and Patrick Morton, *Algebraic dynamics of polynomial maps on the algebraic closure of a finite field, I*, Rocky Mountain J. Math. **24** (1994), no. 2, 453–481.

Note the order the data is presented in, and the different typefaces.

The references are sorted alphabetically according to the authors. Multiple publications are ordered in chronological order.

Use the same last name followed by a reference number used when referring to it.

Example: If we change the previous example to:

```
\begin{thebibliography}{Asc00}
\bibitem[Asc00]{Aschbacher} M. Aschbacher, ...
\bibitem[BM94]{BatraMorton} Anjula Batra ...
\end{thebibliography}
```

we get the following result:

. . . Aschbacher [Asc00] . . . Batra and Morton [BM94] . . .

Bibliography

M. Aschbacher, *Finite Group Theory*, second edition, Cambridge University Press, Cambridge, 2000.

Anjula Batra and Patrick Morton, *Algebraic dynamics of polynomial maps on the algebraic closure of a finite field, I*, Rocky Mountain J. Math. **24** (1994), no. 2, 453–481.

A large bibliography can be put into an external file using BibTEX. Instead of using

```
\begin{bibliography}
...
\end{bibliography}
```

we can use the following commands:

```
\bibliographystyle{IEEEtran}
\bibliography{references}
```

This will parse the complete file *references.bib* for BibTEXentries and automatically add all entries that were cited in the document.

This method is useful for keeping and reusing a large bibliography along several documents.

The file *references.bib* for the previous examples looks like this:

```
@book{Aschbacher2000,
author = {M. Aschbacher},
title = {Finite {G}roup {T}heory},
edition = {2nd},
publisher = {Cambridge {University} {Press}},
address = {Cambridge},
year = {2000}
}

@article{Batra1994,
author = {Anjula Batra and Patrick Morton},
title = {Algebraic dynamics of polynomial maps on
        the algebraic closure of a finite field, {I}},
year = {1994},
journal = {Roky Mountain J. Math.},
volume = 24,
number = 2,
pages = {453--481}
}
```

When using BibTEX some additional steps have to be taken when compiling the document (if your editor of choice does not do this automatically)

```
pdflatex myDocument.tex
bibtex myDocument
pdflatex myDocument.tex
```

The first run of `pdflatex` creates the *.aux* file needed for BibTEX to run. `bibtex myDocument` parses the *.aux* file and creates the file *myDocument.bbl*. The second run of *pdflatex* parses this *.bbl* file and adds the references to the document. You might have to run `pdflatex` a third time for all references to appear correctly.

Typesetting Program Code

Typesetting Program Code

A very good package when it comes to typesetting program code is `listings`. We will study how to get nicely looking Ada-code through some examples marking keywords in bold typeface.

We start by loading the necessary packages and declare a style to use:

```
\usepackage{listings,color}
\lstdefinestyle{Adastyle}{%
  language=Ada,
  columns=fullflexible,
  basicstyle=\small,
  extendedchars=true,
  xleftmargin=2em,
  xrightmargin=2em,
  backgroundcolor=\color[rgb]{0.9,0.9,0.98},
  frameshape={YYY}{n}{n}{YYY}
}
```

Now, we declare an environment that uses the style we defined:

```
\lstnewenvironment{Adacode}[1][]{%
  \lstset{style=Adastyle,#1}}{}
```

Explanation of `#1` is found in the section about programming.

Both options and the number of programming languages that `listings` supports is large. For more information refer to the manual of the package.

Example: The code

```
\begin{Adacode}[caption=Some algorithm.]
J := 0;
while J < 6 loop
  PUT(J);
  J := J + 2;
end loop;
\end{Adacode}
```

gives the result:

Programkod 1: Some algorithm.

```
J := 0;
while J < 6 loop
  PUT(J);
  J := J + 2;
end loop;
```

Example: The result of the code below can be seen on the next page.

```
\begin{Adacode}%[caption=The $n$th Fibonacci number.]
function FIBONACCI(N : POSITIVE) return POSITIVE
begin
  if N = 1 or N = 2 then
    return 1;
  else
    return FIBONACCI(N - 2) + FIBONACCI(N - 1);
  end fi;
end FIBONACCI;
\end{Adacode}
```

To insert, let us say, the lines 54–171 from the file `prog.ada` without having to paste it directly into the code of the document, you may write this:

```
\lstinputlisting[style=Adastyle,
  firstline=54,lastline=171]{prog.ada}
```

```
function FIBONACCI(N : POSITIVE) return POSITIVE
begin
  if N = 1 or N = 2 then
    return 1;
  else
    return FIBONACCI(N − 2) + FIBONACCI(N − 1);
  end fi;
end FIBONACCI;
```

Presentations

Presentations

There are a number of packages and classes for presentations. We briefly show how to use the `beamer` class. Title page:

```
\documentclass{beamer}
\usetheme{PaloAlto}
\title{Elementary number theory}
\author{Robert Nyqvist}
\date{Autumn 2005}
\begin{document}
  \begin{frame}
      \titlepage
  \end{frame}
\end{document}
```

Note that there are a number of different themes to choose from.

Elementary Number Theory

Robert Nyqvist

Autumn 2005

A page:

```
\begin{frame}
  \frametitle{Multiplicative function}
  \begin{definition}
    A \alert{multiplicative function} is an arithmetic
    function $f$ such that if $(m, n) = 1$, then
    $f(mn) = f(m)f(n)$.
  \end{definition}
  \begin{theorem}
    Let $f$ be a multiplicative function and $n =
    p_1^{a_1} \cdots p_n^{a_n}$.  Then $f(n) = f(p_1^{a_1})
    \cdots f(p_n^{a_n})$.
  \end{theorem}
\end{frame}
```

Multiplicative function

Definition

A multiplicative function is an arithmetic function f such that if $(m, n) = 1$, then $f(mn) = f(m)f(n)$.

Theorem

Let f be a multiplicative function and $n = p_1^{a_1} \cdots p_n^{a_n}$. Then $f(n) = f(p_1^{a_1}) \cdots f(p_n^{a_n})$.

Programming

Programming

To define your own commands you use:

\newcommand{*new-command*}[n]{*code*}

where n gives the number of arguments to the command *new-command*, with a maximum of nine. In *code* you refer to the first argument using #1, and to the second using #2 etc.

Note that you may not define an existing command using `\newcommand`. Instead, use `\renewcommand`. But be careful with which commands you redefine.

There are also commands that define new environments or redefine existing environments: `\newenvironment` and `\renewenvironment`, respectively.

Example: Suppose we write the following:

```
\newcommand{\eg}{e\kern.1em g}
```

where `\kern.1em` means that a space of 10% of 1 em is inserted between “e” and “g”.

If we instead want `\eg` should be e.g., we change the code above to:

```
\newcommand{\eg}{e.g.}
```

and the change occurs in all places `\eg` is used.

Example: We create a command that accepts one argument:

```
\newcommand{\good}[1]{Good #1!}
```

EXAMPLE: `\good{heavens}` $\rightarrow$ Good heavens!

EXAMPLE: `\good{morning}` $\rightarrow$ Good morning!

Example: Finally we create a command that accepts two arguments:

```
\newcommand{\tuple}[2]{#1_1, #1_2, \ldots, #1_{#2}}
```

Note that this command may only be used in mathematics mode.

EXAMPLE: `$\tuple{x}{n}$` → $x_1, x_2, \ldots, x_n$

EXAMPLE: `$\tuple{\alpha}{15}$` → $\alpha_1, \alpha_2, \ldots, \alpha_{15}$

Iterations and selections is possible in LaTeX. An appropriate package to easily implement such is `ifthen`, that gives access to these commands:

`\ifthen{`*test*`}{`*if-code*`}{`*else-code*`}`
`\whiledo{`*test*`}{`*code*`}`

Furthermore, you need to know what counters, lengths and boxes are and how they work if you want to program complicated things in LaTeX.

Error Handling

Error Handling

Sometimes there are errors discovered during compilation. When this happens, LaTeX stops and gives a message about what went wrong and asks for new instructions. For instance, it may look like this in the terminal window:

```
! Undefined control sequence.
l.4 \en
      {document}
?
```

that means that the command `\en` is unknown to LaTeX and that the error occurs on row 4 in the text file.

When a warning occurs the compiler does not stop, it simply generates a message. A typical warning may look like this:

```
Overfull \hbox (38.34589pt too wide) in paragraph
at lines 169--171
```

Which, in this case, means that LaTeX was not able to hyphenate a word and that this word is written about 38.34 points into the margin, where a point corresponds to 0,351 mm.

Another common warning is: `Underfull \hbox` – which means that LaTeX had to insert too much space between words on a line.

Some common errors:

- Misspelling a command.
- Forgetting to close braces in complicated expressions.
- Using a command out of order, e.g., `\alpha` in text mode.
- Ending an environment or ending nested environments in the wrong order.
- Placing `\label` before `\caption`.

Tips

Tips

- Start writing as much as possible in LaTeX.
- Create and refine templates with often used custom commands and structure.
- A good work flow: Write a few paragraphs, save, compile, correct possible errors, recompile if necessary, review the result and repeat.
- Focus on content. When following IMRaD, write your parts in this order: Results -> Methods -> Discussion -> Introduction.
- Only do the cosmetics as a last step, when your body of text is proofread and overall set in stone.
- Use the wealth of knowledge found online. *The TeX Catalogue Online* is a good list over different packages for LaTeX.