LaTeX for Beginners

by gracious permission of

Francesco Gambato Ph.D.
francesco.gambato@gmail.com
The Challenge

• The aim is to introduce a beginner to LaTeX and teach the basic commands, so that they can create a simple document and find out whether LaTeX will be useful to them.

• Generate interest and curiosity about a wonderful text creation system, and not only for scientific publications.

• Improve your skills.
What is TeX?

• TeX is a low-level markup and programming language created by Donald Knuth to typeset documents attractively and consistently. Knuth started writing the TeX typesetting engine in 1977 to explore the potential of the digital printing equipment that was beginning to infiltrate the publishing industry at that time, especially in the hope that he could reverse the trend of deteriorating typographical quality that he saw affecting his own books and articles.

• Programming in TeX generally progresses along a very gradual learning curve, requiring a significant investment of time to build custom macros for text formatting. Fortunately, document preparation systems based on TeX, consisting of collections of pre-built macros, do exist.

• One of the most popular macro packages is called LaTeX.
What is LaTeX ?

• LaTeX is not a word processor!

• LaTeX (pronounced either "Lah-tech" or "Lay-tech") is a macro package based on TeX created by Leslie Lamport. Its purpose is to simplify TeX typesetting, especially for documents containing mathematical formulae.

• This approach has some disadvantages in comparison with a WYSIWYG (What You See Is What You Get) program such as Openoffice.org Writer or Microsoft Word. ............maybe not.........

• The LaTeX-like approach can be called WYSIWYM, i.e. What You See Is What You Mean: you can't see what the final version will look like while typing. Instead you see the logical structure of the document. LaTeX takes care of the formatting for you.
Disadvantages of LaTeX

• You don't (usually) see the final version of the document when editing it.

• You generally need to know the necessary commands for LaTeX markup.

• It can sometimes be difficult to obtain a certain look for the document.
Advantages of LaTeX

• You can **concentrate** purely on the **structure and contents** of the document, not get caught up with superficial **layout issues**.

• You **don't need to manually adjust** fonts, text sizes, line heights, or **text flow for readability**, as **LaTeX** takes care of them automatically.

• The layout, fonts, tables and so on are **consistent** throughout the document.

• Mathematical **formulae** can be easily typeset.

• Indexes, footnotes, citations and references are generated easily.

• You are forced to **structure your documents correctly**.
Example

Cartesian closed categories and the price of eggs

Jane Doe
September 1994
Hello world!

To produce this in most typesetting or word-processing systems, the author would have to decide what layout to use, so would select (say) 18pt Times Roman for the title, 12pt Times Italic for the name, and so on. This has two results: authors wasting their time with designs; and a lot of badly designed documents!
Example

LaTeX is based on the idea that it is better to leave document design to document designers, and to let authors get on with writing documents. So, in LaTeX you would input this document as:

\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
  \maketitle
  Hello world!
\end{document}
Example

\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
  \maketitle
  Hello world!
\end{document}

• This document is an article.
• Its \textbf{title} is Cartesian closed categories and the price of eggs.
• Its \textbf{author} is Jane Doe.
• It \textbf{was written} in September 1994.
• The document consists of a \textbf{title} followed by the \textbf{text} Hello world!
Example

Cartesian closed categories and the price of eggs

Jane Doe

September 1994

Hello world!

• This document is an article.
• Its title is Cartesian closed categories and the price of eggs.
• Its author is Jane Doe.
• It was written in September 1994.
• The document consists of a title followed by the text Hello world!
LaTeX - Installation

• LaTeX is not a program by itself, it is a language.

• Using LaTeX requires a bunch of tools.

• LaTeX is available for most computer platforms, since it was programmed to be very portable.

• Linux users may use a command-line interface or a dedicated TeX editor like TeXworks.

• Microsoft Windows users can install an integrated development environment like TeXnicCenter.
LaTeX - Linux

LaTeX is a popular document preparation system that is particularly well-suited for typesetting mathematical and scientific documents. It is based on the TeX typesetting engine and is widely used in academia, especially in fields such as mathematics, physics, and computer science.

Here is a simple example of a LaTeX document that could be used to generate a PDF file:

```latex
\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
\maketitle
Hello world!
\end{document}
```

To compile this document, you can use the `pdflatex` command:

```
dbz@dbz-K5V:--$ pdflatex helloworld.tex
```

This will generate a PDF file named `helloworld.pdf`. The image in the document shows a terminal window with the LaTeX code and the output of the `pdflatex` command.
LaTeX - Linux

\documentclass[article]{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
\maketitle
Hello world!
\end{document}

Cartesian closed categories and the price of eggs

Jane Doe
September 1994

Hello world!
LaTeX - Windows

What is TeXnicCenter?

• TeXnicCenter is an integrated development environment (IDE) used to create, edit and compile LaTeX documents on Microsoft Windows XP, Vista and Windows 7. It includes a code editor with syntax highlighting and provides document structure visualization.

• The compilation process that converts LaTeX documents into a typeset output can be started by simply choosing a menu item or clicking a toolbar button. LaTeX compiler diagnostics are displayed in a separate window allowing to navigate through the errors, warnings and bad box messages.
• A viewer that displays the typeset document can be started using a single click. Viewers that support TeXnicCenter’s forward/inverse search can be used to switch between the typeset document and the corresponding line in the source.

• TeXnicCenter’s aim is to support unexperienced LaTeX users by providing shortcuts to important LaTeX commands via menu entries. Furthermore, the execution of the LaTeX compiler and tools such as MakeIndex and BibTeX is handled automatically. TeXnicCenter supports advanced LaTeX users as well by providing a powerful, fully customizable integrated development environment.
Requirements

step 1

• To typeset LaTeX documents, TeXnicCenter requires a LaTeX distribution to be installed. Major LaTeX distributions supported by TeXnicCenter is MiKTeX.

• MiKTeX (pronounced *mick-tech*) is an up-to-date implementation of TeX/LaTeX and related programs for Windows (all current variants).

• MiKTeX is open source.

• Download MiKTeX:
  http://miktex.org/download
Easy to install

It is very easy to install MiKTeX. The MiKTeX Setup Wizard guides you through the installation process. You don't have to be a computer expert.
Easy to update

To start the wizard, click "Update" in the Windows start menu.

Choose “I want to get updated packages from a remote package repository”, if you want to download updates from the Internet.
Selecting packages

You will presented a list of updateable packages.

Click "Next", to start the update process.
Update progress

When the update process has finished, you can click "Next" to go to the last page.
Finish

The selected updates have been installed.
Error package

Don't panic if you see error messages like this.
First Steps with LaTeX

To start the wizard, click "Package Manager" in the Windows start menu.
Error package

Select the package and install by +

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Size</th>
<th>Installed on</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>ametoc</td>
<td>\Uncategorized</td>
<td>818873</td>
<td>2008-09-05</td>
<td>A classical Arabic typeface, Neskh style</td>
</tr>
<tr>
<td>amitii</td>
<td>\Uncategorized</td>
<td>396241</td>
<td>2014-01-06</td>
<td></td>
</tr>
<tr>
<td>amssadr</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>204376</td>
<td>2012-04-08</td>
<td>Alter the position of affiliations in amssart</td>
</tr>
<tr>
<td>amscls</td>
<td>\Formats\LaTeX\Basic \LaTeX</td>
<td>1461022</td>
<td>2012-02-01</td>
<td>AMS document classes for \LaTeX</td>
</tr>
<tr>
<td>amsfonts</td>
<td>\Fonts\METAFONT Fonts</td>
<td>7932891</td>
<td>2009-09-03</td>
<td>2012-04-04 TeX fonts from the American Mathematical Society</td>
</tr>
<tr>
<td>amstex</td>
<td>\Uncategorized</td>
<td>2647862</td>
<td>2009-07-13</td>
<td>Miscellaneous \LaTeX enhancements</td>
</tr>
<tr>
<td>amstex-primer</td>
<td>\Uncategorized</td>
<td>5726962</td>
<td>2012-02-01</td>
<td>Getting up and running with AMS-\LaTeX</td>
</tr>
<tr>
<td>amsmath</td>
<td>\Formats\LaTeX\Basic \LaTeX</td>
<td>2584083</td>
<td>2013-03-11</td>
<td>AMS mathematical facilities for \LaTeX</td>
</tr>
<tr>
<td>amssref</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>2413736</td>
<td>2013-03-11</td>
<td>A \LaTeX-based replacement for BibTeX</td>
</tr>
<tr>
<td>amstex</td>
<td>\Applications\Math</td>
<td>637694</td>
<td>2011-06-23</td>
<td>American Mathematical Society plain TeX macros</td>
</tr>
<tr>
<td>analogclock</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>97337</td>
<td>2008-12-23</td>
<td>An analog ticking clock package for PDF output</td>
</tr>
<tr>
<td>animate</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>3925729</td>
<td>2012-09-27</td>
<td>Create PDF animations from graphics files and inline graphics</td>
</tr>
<tr>
<td>annot-pro</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>2114579</td>
<td>2012-11-22</td>
<td>Create text, stamp, and file attachment annotations</td>
</tr>
<tr>
<td>anonchap</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>195244</td>
<td>2010-02-22</td>
<td>Make chapters be typeset like sections</td>
</tr>
<tr>
<td>anonymouspro</td>
<td>\Uncategorized</td>
<td>1487435</td>
<td>2014-01-06</td>
<td>Use AnonymousPro fonts with \LaTeX</td>
</tr>
<tr>
<td>answers</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>244833</td>
<td>2010-10-16</td>
<td>Setting questions (or exercises) and answers</td>
</tr>
<tr>
<td>antique</td>
<td>\Fonts\Outline fonts</td>
<td>146243</td>
<td>2004-08-15</td>
<td></td>
</tr>
<tr>
<td>antp</td>
<td>\Language Support\Polish</td>
<td>331056</td>
<td>2001-05-31</td>
<td>Antykowa Pi\’o\’l\’skiego: a Type 1 family of Polish traditional type</td>
</tr>
<tr>
<td>antyttoor</td>
<td>\Language Support\Polish</td>
<td>23628036</td>
<td>2008-09-13</td>
<td>Antykowa Toru\’nska: a Type 1 family of a Polish traditional type</td>
</tr>
<tr>
<td>anufinalexam</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>5278</td>
<td>2010-12-13</td>
<td>\LaTeX document shell for ANU final exam</td>
</tr>
<tr>
<td>anyfontsiz e</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>5695</td>
<td>2007-12-09</td>
<td>Select any font size in \LaTeX</td>
</tr>
<tr>
<td>anysize</td>
<td>\Formats\LaTeX\LaTeXe contrib</td>
<td>11157</td>
<td>2001-11-28</td>
<td></td>
</tr>
<tr>
<td>aspect</td>
<td>\Uncategorized</td>
<td>948332</td>
<td>2014-01-14</td>
<td>TikZ styles for creating overlaid pictures in beamer</td>
</tr>
</tbody>
</table>
Requirements step 2

• To view typeset PDF documents, a PDF viewer should be installed as well. The recommended PDF viewer is **Sumatra PDF** which provides an excellent forward/inverse search support.

• Sumatra PDF is a free PDF reader for Windows.

• Sumatra PDF is powerful, small, portable and starts up very fast.

• Download Sumatra PDF: http://blog.kowalczyk.info/software/sumatrapdf/download-free-pdf-viewer.html
Requirements
step 3

- TeXnicCenter is what we call an integrated documentation environment (IDE) for LaTeX.

- That means that TeXnicCenter integrates all the functionality you need to create, write, build, fix, view and print your LaTeX documents.

- TeXnicCenter is free and open source (GPL).

- Download: http://www.texniccenter.org
Summarizing

• step 1
• step 2
• step 3
LaTeX syntax - markup

LaTeX uses a markup language in order to describe document structure and presentation. LaTeX converts your source text, combined with the markup, into a high quality document. For the purpose of analogy, web pages work in a similar way: the HTML is used to describe the document, but it is your browser that presents it in its full glory - with different colours, fonts, sizes, etc.

\documentclass{article}
\begin{document}
Hello world!
\end{document}
LaTeX syntax - spaces

The LaTeX compiler normalises whitespace so that whitespace characters, such as `[space]` or `[tab]`, are treated uniformly as "space": several consecutive "spaces" are treated as one, "space" opening a line is generally ignored, and a single line break also yields “space”. A double line break (an empty line), however, defines the end of a paragraph; multiple empty lines are also treated as the end of a paragraph.

An example of applying these rules is presented below: the left-hand side shows the user's input (.tex), while the right-hand side depicts the rendered output (.dvi/.pdf/.ps).

It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.

It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.
LaTeX syntax – reserved characters

The following symbols are reserved characters that either have a special meaning under LaTeX or are unavailable in all the fonts. If you enter them directly in your text, they will normally not print, but rather make LaTeX do things you did not intend.

\# $ \% ^ \& _ { } \sim \backslash

As you will see, these characters can be used in your documents all the same by adding a prefix backslash:

\# \$ \% \^ \& \_ \{ \} \sim \textbackslash

The backslash character \ cannot be entered by adding another backslash in front of it (\\); this sequence is used for line breaking. For introducing a backslash in math mode, you can use \textbackslash instead.
LaTeX syntax – groups

A group is basically defined by a pair of braces. The range of commands put between braces is limited to them. The `\begingroup` and `\endgroup` commands are equivalent to opening brace and closing brace.

For some commands it is important to restrict their range of action, and that's where groups come to be very useful.

```latex
{ 
  \bf This is bold. 
}
This is no longer bold.
```
LaTeX syntax – environments

Environments in LaTeX have a role that is quite similar to commands, but they usually have effect on a wider part of the document. Their syntax is:

\begin{environmentname}
text to be influenced
\end{environmentname}

Between the \begin and the \end you can put other commands and nested environments. The internal mechanism of environments defines a group, which makes its usage safe (no influence on the other parts of the document). In general, environments can accept arguments as well, but this feature is not commonly used.

Anything in LaTeX can be expressed in terms of commands and environments.
LaTeX syntax – commands

LaTeX commands are case sensitive, and take one of the following two formats:

• They start with a backslash \ and then have a name consisting of letters only. Command names are terminated by a space, a number or any other "non-letter".
• They consist of a backslash \ and exactly one non-letter.

Some commands need an argument, which has to be given between curly braces { } after the command name. Some commands support optional parameters, which are added after the command name in square brackets [ ]. The general syntax is:

```
\commandname[option1,option2,...]{argument1}{argument2}...
```
LaTeX syntax – commands - switches

Most standard LaTeX commands have a switch equivalent. Switches have no arguments but apply on the rest of the scope, i.e. the current group or environment. A switch should (almost) never be called outside of any scope, otherwise it will apply on the rest of the document.

% \emph is a command with argument, \em is a switch. 
\emph{emphasized text}, this part is normal % Correct
{\em emph emphasized text}, this part is normal % Correct

\em emphasized text, this part is normal % Incorrect
{\em emph{emphasized text}}, this part is normal % Incorrect

emphasized text, this part is normal
emphasized text, this part is normal
emphasized text, this part is normal
emphasized text, this part is normal

Correct
Incorrect
LaTeX syntax – comments

When LaTeX encounters a `%` character while processing an input file, it ignores the rest of the current line, the line break, and all whitespace at the beginning of the next line. This can be used to write notes into the input file, which will not show up in the printed version.

This is an `% stupid
% Better: instructive <----
example: Supercal%
    ifragilist%
icexpialidocious

This is an example: Supercalifragilisticexpialidocious
Example

% hello.tex – Our first LaTeX example!
\documentclass{article}
\begin{document}
Hello World!
\end{document}

Hello World!

% hello.tex – Our first LaTeX example!

The first line is a comment. This is because it begins with the percent symbol (\%); when LaTeX sees this, it simply ignores the rest of the line. Comments are useful for people to annotate parts of the source file. For example, you could put information about the author and the date, or whatever you wish.
Example

% hello.tex - Our first LaTeX example!
\documentclass{article}
\begin{document}
  Hello World!
\end{document}

\documentclass{article}

This line is a command and tells LaTeX to use the article document class. A document class file defines the formatting, which in this case is a generic article format. The handy thing is that if you want to change the appearance of your document, substitute article for another class file that exists.
## Example

% hello.tex - Our first LaTeX example!
\documentclass{article}
\begin{document}
  Hello World!
\end{document}

This line is the beginning of the environment called `document`; it alerts LaTeX that content of the document is about to commence. Anything above this command is known generally to belong in the preamble.
LaTeX/Introduction

First Steps with LaTeX

Example

% hello.tex - Our first LaTeX example!
\documentclass{article}
\begin{document}
Hello World!
\end{document}

Hello World!

This was the only actual line containing real content - the text that we wanted displayed on the page.
LaTeX/Introduction

Example

% hello.tex - Our first LaTeX example!
documentclass{article}
begin{document}
  Hello World!
end{document}

\end{document}

Hello World!

The document environment ends here. It tells LaTeX that the document source is complete, anything after this line will be ignored.
LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical-scientific documents but it can be used for almost any format publishing.

LaTeX is not a word processor! Instead, LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content: For example, consider this document.

To produce this in most typesetting or word-processing systems, the author would have to decide what layout to use, so would select (say) 13pt Times Roman for the title, 12pt Times Italic for the name, and so on. This has two results: authors wasting their time with designs; and a lot of badly designed documents!

LaTeX is based on the idea that it is better to leave document design to document designers, and to let authors get on with writing documents. So, in LaTeX you would input this document as:

Table des matières

LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical-scientific documents but it can be used for almost any format publishing.

LaTeX is not a word processor! Instead, LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content: For example, consider this document.

To produce this in most typesetting or word-processing systems, the author would have to decide what layout to use, so would select (say) 13pt Times Roman for the title, 12pt Times Italic for the name, and so on. This has two results: authors wasting their time with designs; and a lot of badly designed documents!

LaTeX is based on the idea that it is better to leave document design to document designers, and to let authors get on with writing documents. So, in LaTeX you would input this document as:

**Figure 1.** The aq-aq coordinate system

This is a diagram which explains the altitude-azimuth, or "alt-az", coordinate system. It is a purely local system: the coordinates determined by an observer in New York will not agree with those determined simultaneously by an observer in Texas.

Notice that, in the final typeset document, the diagram doesn’t always end up in the same place relative to the surrounding text as it does in the input file.
Global Structure

When LaTeX processes an input file, it expects it to follow a certain structure. Thus every input file must contain the commands:

\documentclass{...}
\usepackage{...}

\begin{document}
...
\end{document}

The area between \documentclass{...} and \begin{document} is called the preamble. It normally contains commands that affect the entire document.
Global Structure

After the preamble, the text of your document is enclosed between two commands which identify the beginning and end of the actual document:

\begin{document}
...
\end{document}

You would put your text where the dots are. The reason for marking off the beginning of your text is that LaTeX allows you to insert extra setup specifications before it (where the blank line is in the example above: we'll be using this soon). The reason for marking off the end of your text is to provide a place for LaTeX to be programmed to do extra stuff automatically at the end of the document (a storage), like making an index.
Preamble

When processing an input file, LaTeX needs to know the type of document the author wants to create. This is specified with the `\documentclass` command. It is recommended to put this declaration at the very beginning.

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

Here class specifies the type of document to be created. The LaTeX distribution provides additional classes for other documents, including letters and slides. It is also possible to create your own, as is often done by journal publishers, who simply provide you with their own class file, which tells LaTeX how to format your content. But we'll be happy with the standard article class for now. The options parameter customizes the behavior of the document class. The options have to be separated by commas.
Example

An input file for a LaTeX document could start with the line:

\documentclass[11pt,twoside,a4paper]{article}

which instructs LaTeX to typeset the document as an article with a base font size of 11 points, and to produce a layout suitable for double sided printing on A4 paper.
## Example

### some Document Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>article</td>
<td>For articles in scientific journals, presentations, short reports, program documentation, invitations, ...</td>
</tr>
<tr>
<td>report</td>
<td>For longer reports containing several chapters, small books, thesis, ...</td>
</tr>
<tr>
<td>book</td>
<td>For real books.</td>
</tr>
<tr>
<td>letter</td>
<td>For writing letters.</td>
</tr>
<tr>
<td>beamer</td>
<td>For writing presentations.</td>
</tr>
</tbody>
</table>
**Example**

<table>
<thead>
<tr>
<th>some Document Classes Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>10pt, 11pt, 12pt</strong></td>
</tr>
<tr>
<td><strong>a4paper, letterpaper,...</strong></td>
</tr>
<tr>
<td><strong>titlepage, notitlepage</strong></td>
</tr>
<tr>
<td><strong>twoside, oneside</strong></td>
</tr>
</tbody>
</table>
Example

For example, if you want a report to be in 12pt type on A4, but printed one-sided, you would use:

\documentclass[12pt,a4paper,twoside]{report}
Packages

While writing your document, you will probably find that there are some areas where basic LaTeX cannot solve your problem. If you want to include graphics, colored text or source code from a file into your document, you need to enhance the capabilities of LaTeX. Such enhancements are called packages. Some packages come with the LaTeX base distribution. Others are provided separately. Modern TeX distributions come with a large number of packages pre-installed. Packages are activated with the

\usepackage[options]{package}

command, where package is the name of the package and options is a list of keywords that trigger special features in the package.
Example

For example, to use the `color` package, which lets you typeset in colors, you would type:

\begin{document}
\documentclass[11pt,a4paper,oneside]{report}
\usepackage{color}
\begin{document}
\begin{document}
\end{document}
\end{document}
Packages

You can include several package names in one `\usepackage` command by separating the names with commas, like this:

\usepackage{package1, package2, package3}
Packages

You can have more than one \texttt{\usepackage} command. Some packages allow optional settings in square brackets. If you use these, you must give the package its own separate \texttt{\usepackage} command, like geometry shown below:

\begin{verbatim}
\documentclass[11pt,a4paper,oneside]{report}
\usepackage{pslatex,palatino,avant,graphicx,color}
\usepackage[margin=2cm]{geometry}

\begin{document}
\title{\textcolor{red}{Practical Typesetting}}
\author{\textcolor{blue}{Name}\ \ \ \ Work}
\date{\textcolor{green}{December 2005}}
\maketitle

\end{document}
\end{verbatim}
Packages

You can have more than one \usepackage command. Some packages allow optional settings in square brackets. If you use these, you must give the package its own separate \usepackage command, like geometry shown below:

★

Practical Typesetting

Name
Work

December 2005
Packages

Many packages can have additional formatting specifications in optional arguments in square brackets, in the same way as geometry does. Read the documentation for the package concerned to find out what can be done. You can pass several options together separated by a comma:

\usepackage[option1,option2,option3]{package_name}
Top matter

At the beginning of most documents there will be information about the document itself, such as the title and date, and also information about the authors, such as name, address, email etc. All of this type of information within LaTeX is collectively referred to as top matter. Although never explicitly specified (there is no \topmatter command) you are likely to encounter the term within LaTeX documentation.

\documentclass[11pt,a4paper]{report}

\begin{document}
\title{How to Structure a LaTeX Document}
\author{Andrew Roberts}
\date{December 2004}
\maketitle
Hello world!
\end{document}
Top matter

The `\title, \author, and \date` commands are self-explanatory. You put the title, author name, and date in curly braces after the relevant command. The title and author are usually compulsory (at least if you want LaTeX to write the title automatically).

How to Structure a LaTeX Document

Andrew Roberts

December 2004
Top matter

You always finish the top matter with the \texttt{\textbackslash maketitle} command, which tells LaTeX that it's complete and it can typeset the title according to the information you have provided and the class (style) you are using.

\begin{center}
\textbf{How to Structure a LaTeX Document}
\end{center}

\begin{center}
Andrew Roberts
\end{center}

\begin{center}
December 2004
\end{center}
If you omit `\maketitle`, the titling will never be typeset (unless you write your own).

Hello world!
Abstract

As most research papers have an abstract, there are predefined commands for telling LaTeX which part of the content makes up the abstract. This should appear in its logical order, therefore, after the top matter, but before the main sections of the body. This command is available for the document classes article and report, but not book.

\documentclass{article}

\begin{document}

\begin{abstract}
Your abstract goes here...
\end{abstract}

\end{document}
Abstract

By default, LaTeX will use the word "Abstract" as a title for your abstract. If you want to change it into anything else, e.g. "Executive Summary", add the following line before you begin the abstract environment:

\renewcommand{\abstractname}{Executive Summary}
Table of contents

All auto-numbered headings get entered in the Table of Contents (ToC) automatically. You don't have to print a ToC, but if you want to, just add the command \tableofcontents at the point where you want it printed (usually after the Abstract or Summary).

Entries for the ToC are recorded each time you process your document, and reproduced the next time you process it, so you need to re-run \LaTeX one extra time to ensure that all ToC page number references are correctly calculated.
Sectioning commands

The commands for inserting sections are fairly intuitive. Of course, certain commands are appropriate to different document classes. For example, a book has chapters but an article doesn't. Here are some of the structure commands:

\section{Introduction}
This section's content...

\section{Structure}
This section's content...

\subsection{Top Matter}
This subsection's content...

\subsubsection{Article Information}
This subsubsection's content...
Sectioning commands

• Notice that you do not need to specify section numbers.

• LaTeX will sort that out for you.

• Also, for sections, you do not need to markup which content belongs to a given block, using \begin and \end commands.

• LaTeX provides 7 levels of depth for defining sections (see next table). Each section in this table is a subsection of the one above it.

• All the titles of the sections are added automatically to the table of contents (if you decide to insert one).
# Sectioning commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Level</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>\part{xxx}</td>
<td>-1</td>
<td>not in letters</td>
</tr>
<tr>
<td>\chapter{xxx}</td>
<td>0</td>
<td>only books and reports</td>
</tr>
<tr>
<td>\section{xxx}</td>
<td>1</td>
<td>not in letters</td>
</tr>
<tr>
<td>\subsection{xxx}</td>
<td>2</td>
<td>not in letters</td>
</tr>
<tr>
<td>\subsubsection{xxx}</td>
<td>3</td>
<td>not in letters</td>
</tr>
<tr>
<td>\paragraph{xxx}</td>
<td>4</td>
<td>not in letters</td>
</tr>
<tr>
<td>\subparagraph{xxx}</td>
<td>5</td>
<td>not in letters</td>
</tr>
</tbody>
</table>
Sectioning commands

But if you make manual styling changes to your heading, for example a very long title, or some special line-breaks or unusual font-play, this would appear in the Table of Contents as well, which you almost certainly don't want. LaTeX allows you to give an optional extra version of the heading text which only gets used in the Table of Contents and any running heads, if they are in effect. This optional alternative heading goes in [square brackets] before the curly braces:

\section[Effect on staff turnover]{An analysis of the effect of the revised recruitment policies on staff turnover at divisional headquarters}
Sectioning commands

Contents

1 Editing compile
   1.1 First Compile ................................................. 2
   1.1.1 Output formats ............................................. 2

2 Document Structure
   2.1 Reserved Characters ......................................... 3
   2.1.1 Effect on staff turnover ................................. 3
Sectioning commands

Chapter 1

Editing compile

1.1 First Compile

how to compile basic hello world into a pdf. Write your favorite text editor create file and copy/paste the following (with hello.tex):

1.1.1 Output formats

different output formats (dvi, pdf) The output of this command $latex hello.tex will be a dvi file (hello.dvi). This file (.dvi) can be converted by $dvipdf hello.dvi The get an pdf file from tex file, run this command $texi2pdf hello.tex
Sectioning commands

★ Chapter 2

Document Structure

2.1 Reserved Characters

The following symbols characters are reserved by LATEX because they introduce a command and have a special meaning.

2.1.1 An analysis of the effect of the revised recruitment policies on staff turnover at divisional headquarters

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
Section numbering

Numbering of the sections is performed automatically by LaTeX, so don't bother adding them explicitly, just insert the heading you want between the curly braces. Parts get roman numerals (Part I, Part II, etc.); chapters and sections get decimal numbering like this document, and appendices (which are just a special case of chapters, and share the same structure) are lettered (A, B, C, etc.).

To get an unnumbered section heading which does not go into the Table of Contents, follow the command name with an asterisk before the opening curly brace:

\subsection*{Introduction}
Section numbering

Contents

1 Editing compile
   1.1 First Compile ......................................................... 2

2 Document Structure
   2.1 Reserved Characters ................................................. 3
Section numbering

★ Chapter 1

Editing compile

1.1 First Compile

how to compile basic hello world into a pdf. Write your favorite text editor create file and copy/paste the following (with hello.tex):

Output formats

different output formats (dvi, pdf) The output of this command $latex hello.tex will be a dvi file (hello.dvi). This file (.dvi) can be converted by $dvipdf hello.dvi The get an pdf file from tex file, run this command $texpdf hello.tex
Section numbering

★ Chapter 2

Document Structure

2.1 Reserved Characters

The following symbols characters are reserved by LATEX because they introduce a command and have a special meaning.

Effect on staff turnover

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
Section numbering

All the divisional commands from \part* to \subparagraph* have this "starred" version which can be used on special occasions for an unnumbered heading.

If you want the unnumbered section to be in the table of contents anyway, use the \addcontentsline command like this:

\section*{Introduction}
\addcontentsline{toc}{section}{Introduction}
Section numbering

Contents

1  Editing compile
   1.1  First Compile ..............................................
   Output formats ..............................................

2  Document Structure
   2.1  Reserved Characters ......................................
   Effect on staff turnover ....................................

2
2
2
3
3
3
Section numbering

★ Chapter 1

Editing compile

1.1 First Compile

how to compile basic hello world into a pdf. Write your favorite text editor create file and copy/paste the following (with hello.tex):

Output formats
different output formats (dvi, pdf) The output of this command $latex hello.tex will be a dvi file (hello.dvi). This file (.dvi) can be converted by $dvipdf hello.dvi The get an pdf file from tex file, run this command $texi2pdf hello.tex
Section numbering

Chapter 2

Document Structure

2.1 Reserved Characters

The following symbols characters are reserved by LATEX because they introduce a command and have a special meaning.

Effect on staff turnover

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla
List structures

Convenient and predictable list formatting is one of the many advantages of using LaTeX.

Lists often appear in documents, especially academic, as their purpose is often to present information in a clear and concise fashion. List structures in LaTeX are simply environments which essentially come in three flavors: itemize, enumerate and description.
List structures

All lists follow the basic format:

\begin{list_type}
  \item The first item
  \item The second item
  \item The third etc \ldots
\end{list_type}

All three of these types of lists can have multiple paragraphs per item: just type the additional paragraphs in the normal way, with a blank line between each. So long as they are still contained within the enclosing environment, they will automatically be indented to follow underneath their item.
List structures - \texttt{itemize}

This environment is for your standard bulleted list of items.

\begin{itemize}
\item The first item
\item The second item
\item The third etc \ldots
\end{itemize}
List structures - \texttt{enumerate}

The \texttt{enumerate} environment is for ordered lists, where by default, each item is numbered sequentially.

\begin{enumerate}
    \item The first item
    \item The second item
    \item The third etc \ldots
\end{enumerate}

\begin{itemize}
    \item 1. The first item
    \item 2. The second item
    \item 3. The third etc \ldots
\end{itemize}
List structures - \texttt{description}

The description environment is slightly different. You can specify the item label by passing it as an optional argument (although optional, it would look odd if you didn't include it!). Ideal for a series of definitions, such as a glossary.

\begin{description}
  \item[First] The first item
  \item[Second] The second item
  \item[Third] The third etc
  \ldots
\end{description}

\textbf{First} The first item
\textbf{Second} The second item
\textbf{Third} The third etc ...
Sometimes you want a description where the text begins on a new line. This cannot easily be done with `\`. The trick is to use `\hfill`.

\begin{description}
  \item[First] \hfill \\ \\
  The first item
  \item[Second] \hfill \\ \\
  The second item
  \item[Third] \hfill \\ \\
  The third etc \ldots
\end{description}
List structures - nested lists

LaTeX will happily allow you to insert a list environment into an existing one (up to a depth of four—if you need more than four, use the \texttt{easylist} package). Simply begin the appropriate environment at the desired point within the current list. Latex will sort out the layout and any numbering for you.

\begin{enumerate}
\item The first item
  \begin{enumerate}
    \item Nested item 1
    \item Nested item 2
  \end{enumerate}
\item The second item
\item The third etc \ldots
\end{enumerate}

1. The first item
   (a) Nested item 1
   (b) Nested item 2
2. The second item
3. The third etc \ldots
List structures - customizing lists

Customizing LaTeX is outside the beginner’s domain. While not necessarily difficult in itself, because beginners are already overwhelmed with the array of commands and environments, moving on to more advanced topics runs the risk of confusion.

.... good job
\[ k_{n+1} = n^2 + k_n^2 - k_{n-1} \]

\[ \forall x \in X, \exists y \leq \epsilon \]

\[ \lim_{x \to \infty} \exp(-x) = 0 \]

\[ A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & -1 & -3 \\ 1 & 4 & 6 \end{pmatrix}, \quad M = \begin{pmatrix} 1 & 1 & 1 & 2 \\ 2 & -1 & -3 & 5 \\ 1 & 4 & 6 & 1 \\ 1 & -2 & -4 & 3 \end{pmatrix} \]

\[ r(A) \leq 3, \quad (\det M = 0 \implies r(M) \leq 3) \]
Mathematics

One of the greatest motivating forces for Donald Knuth when he began developing the original TeX system was to create something that allowed simple construction of mathematical formulas, while looking professional when printed.

The fact that he succeeded was most probably why TeX (and later on, LaTeX) became so popular within the scientific community.

Typesetting mathematics is one of LaTeX's greatest strengths. It is also a large topic due to the existence of so much mathematical notation.
The `amsmath` and `mathtools` package

If your document requires only a few simple mathematical formulas, plain LaTeX has most of the tools that you will need.

If you are writing a scientific document that contains numerous complicated formulas, the `amsmath` package introduces several new commands that are more powerful and flexible than the ones provided by LaTeX.

The `mathtools` package fixes some `amsmath` quirks and adds some useful settings, symbols, and environments to `amsmath`. 
The \texttt{amsmath} and \texttt{mathtools} package

To use either package, include:

\begin{verbatim}
usepackage{amsmath}
\end{verbatim}

or

\begin{verbatim}
usepackage{mathtools}
\end{verbatim}

in the preamble of the document.

The \texttt{mathtools} package loads the \texttt{amsmath} package and hence there is no need to \texttt{usepackage{amsmath}} in the preamble if \texttt{mathtools} is used.
Mathematics environments

LaTeX needs to know beforehand that the subsequent text does indeed contain mathematical elements.

This is because LaTeX typesets maths notation differently from normal text.

Therefore, special environments have been declared for this purpose. They can be distinguished into two categories depending on how they are presented:

- **text** - text formulas are displayed inline, that is, within the body of text where it is declared.

- **displayed** - displayed formulas are separate from the main text.
Mathematics environments

As maths require special environments, there are naturally the appropriate environment names you can use in the standard way. Unlike most other environments, however, there are some handy shorthands to declaring your formulas. The following table summarizes them:

<table>
<thead>
<tr>
<th>Type</th>
<th>text</th>
<th>displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>math</td>
<td>displaymath</td>
</tr>
<tr>
<td>LaTeX shorthand</td>
<td>(\ldots)</td>
<td>[\ldots]</td>
</tr>
<tr>
<td>TeX shorthand</td>
<td>$\ldots$</td>
<td>$$\ldots$$</td>
</tr>
</tbody>
</table>

**Suggestion:** Using the $$\ldots$$ should be avoided, as it may cause problems.
Mathematics environments

If you are typing text normally, you are said to be in text mode, but while you are typing within one of those mathematical environments, you are said to be in math mode, that has some differences compared to the text mode:

• Most spaces and line breaks do not have any significance, as all spaces are either derived logically from the mathematical expressions, or have to be specified with special commands such as \quad

• Empty lines are not allowed. ----> you will get an error message !!!!

• Each letter is considered to be the name of a variable and will be typeset as such. If you want to typeset normal text within a formula (normal upright font and normal spacing) then you have to enter the text using dedicated commands.
"displayed" maths inside blocks of text

In order for some operators, such as $\lim$ or $\sum$ to be displayed correctly inside some math environments (inline (within text) formulas), it might be convenient to write the \texttt{\scriptsize displaystyle} class inside the environment. Doing so might cause the line to be taller, but will cause exponents and indices to be displayed correctly for some math operators.

For example, the $\sum_{i=1}^{n} a_i = 3$ will print a smaller $\sum_{i=1}^{n} a_i = 3$ and $\texttt{\scriptsize displaystyle} \sum_{i=1}^{n} a_i = 3$ will print a bigger one $\sum_{i=1}^{n} a_i = 3$, like in equations.
Symbols

Mathematics has many symbols!

There are a great deal of examples such as Greek letters, set and relations symbols, arrows, binary operators, etc.

\[ \forall x \in X, \quad \exists y \leq \epsilon \]

\texttt{\ forall \ x \ \in \ X, \ \quad \ \exists \ y \ \leq \ \epsilon}
Operators

An operator is a function that is written as a word: e.g. trigonometric functions (sin, cos, tan), logarithms and exponentials (log, exp). LaTeX has many of these defined as commands:

$$\lim_{x \to \infty} \exp(-x) = 0$$

\[ \lim_{x \to \infty} \exp(-x) = 0 \]
Powers and indices

Powers and indices are equivalent to superscripts and subscripts in normal text mode. The caret (^) character is used to raise something, and the underscore (_) is for lowering. If more than one expression is raised or lowered, they should be grouped using curly braces (\{ and \}).

\[ k_{n+1} = n^2 + k_n^2 - k_{n-1} \]

\[ k\{n+1\} = n^2 + k\_n^2 - k\_{n-1} \]
... You Can Do Anything You Want ...

... only with LaTeX
Equation numbering

The equation environment automatically numbers your equation.

You can also use the \texttt{\textbackslash{}label} and \texttt{\textbackslash{}ref} (or \texttt{\textbackslash{}eqref} from the amsmath package) commands to label and reference equations, respectively. For equation number 1, \texttt{\textbackslash{}ref} results in 1 and \texttt{\textbackslash{}eqref} results in (1).

\begin{align*}
    h^-(X|Y) & \leq \frac{n+1}{e} - h(X|Y) \\
    + \int p(y) \log \left( \frac{E[|X|^2|Y=y]}{n} \right) \, dy & \quad \text{(11)}
\end{align*}

\begin{align*}
    P_U(u) & = \begin{cases} 
        0.1 & \text{if } u = 0, \\
        0.3 & \text{if } u = 1, \\
        0.6 & \text{if } u = 2.
    \end{cases} \\
    x = a + b & \iff \begin{cases} 
        a = \frac{x}{2} + \frac{y}{2} \\
        b = \frac{x}{2} - \frac{y}{2}
    \end{cases} \quad \text{(63)}
\end{align*}
Equation numbering

\begin{equation}
\label{eq:complex number}
  z = \overbrace{\underbrace{x}_\text{real} + \underbrace{iy}_\text{imaginary}}^{\text{complex number}}
\end{equation}

this references the equation \ref{eq:complex number}.

this references the equation \eqref{eq:complex number}.

\begin{equation}
  \begin{array}{c}
    \text{complex number} \\
    \hline
    x \quad + \\[10pt]
    iy
  \end{array}
\end{equation}

(1.1)

this references the equation 1.1.
this references the equation (1.1).
Chemical Graphics

\[
\begin{align*}
\text{H} & \text{C} \text{H} \\
\text{H} & \text{H}
\end{align*}
\]

\[
\begin{align*}
R^1 & \text{OH} \\
R^2 & \\
R^3 & \\
R^4 & \\
\text{N} & \text{N} \\
\text{R} & \text{R}
\end{align*}
\]

\[
\begin{align*}
\text{ROH} + \text{R} \text{C} \text{OOH} & \rightarrow \text{R} \text{C} \text{OR} + \text{H}_2\text{O} \\
\text{Alcohol} & \text{Carboxylic acid} & \text{Ester} & \text{Water}
\end{align*}
\]
The \texttt{chemfig} package

Chemfig is a package used to draw 2D chemical structures.

Chemfig uses the tikz package to produce its graphics and is used by adding the following to the preamble:

\begin{verbatim}
\usepackage{chemfig}
\end{verbatim}

The primary command used in this package is \texttt{\chemfig{}}:

\begin{verbatim}
\chemfig{<atom1><bond type>[<angle>,<coeff>,<tikz code>]<atom2>}
\end{verbatim}

\ldots a few quick examples \ldots
Example

A methane molecule, for instance, can be produced with the following code:

Example

Drawing chemical structures:

\chemfig{
  *6((-(-R^2)=-
    (=-[:::-60]N-}*6(=(-R^3)-=(R^4)-=(R^3)-))
  =(-OH)-(-R^1)=)
}
Example

The molecules can be named:

\begin{verbatim}
\chemname{\chemfig{R’OH}}{Alcohol}
\chemsign{+}
\chemname{\chemfig{R-C(-[::30]OH)=[:30]O}}{Carboxylic acid}
\chemrel{->}
\chemname{\chemfig{R-C(-[::30]OR’)=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_2O}}{Water}
\end{verbatim}

\begin{center}
\includegraphics[width=\textwidth]{example.png}
\end{center}
Data: this text
\textbf{Result: }how to write algorithm with \LaTeX\2e initialization;
\texttt{while not at end of this document do}
  \texttt{read current;}
  \texttt{if understand then}
    \texttt{go to next section;}
    \texttt{current section becomes this one;}
  \texttt{else}
    \texttt{go back to the beginning of current section;}
\texttt{end}
Algorithm 1: How to write algorithms

\begin{align*}
\text{if } i \geq \text{maxval} \text{ then } & i \leftarrow 0 \\
\text{else } & \text{if } i + k \leq \text{maxval} \text{ then } i \leftarrow i + k \\
\text{end if} & \text{end if}
\end{align*}
The \texttt{algorithm2e} package

LaTeX has several packages for typesetting algorithms in form of "pseudocode". They provide stylistic enhancements over a uniform style (i.e., all in typewriter font) so that constructs such as loops or conditionals are visually separated from other text.

The \texttt{algorithm2e} package allows typesetting algorithms with a lot of customization.

The package is loaded like:

\texttt{\usepackage[]\{algorithm2e\}}
Example

\begin{algorithm}[H]
\KwData{this text}
\KwResult{how to write algorithm with \LaTeX\texttt{2e} }
initialization\;
\While{not at end of this document}{
read current\;
\eIf{understand}{
go to next section\;
current section becomes this one\;}
{go back to the beginning of current section\;}
}
\caption{How to write algorithms}
\end{algorithm}
Example

Data: this text
Result: how to write algorithm with \LaTeX2e initialization;
while not at end of this document do
  read current;
  if understand then
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
end
end

Algorithm 1: How to write algorithms
The `algorithmicx` package

The `algorithmicx` package provides a number of popular constructs for algorithm designs.

The package `algorithmicx` itself doesn’t define any algorithmic commands, but gives a set of macros to define such a command set. You may use only `algorithmicx`, and define the commands yourself, or you may use one of the predefined command sets.

Put `\usepackage{algpseudocode}` in the preamble to use the algorithmic environment to write algorithm pseudocode (`\begin{algorithmic}...\end{algorithmic}`).
Example

Below is an example of typesetting a basic algorithm using the \texttt{algorithmicx} package (remember to add the \texttt{\usepackage{algpseudocode}} statement to your document preamble):

\begin{algorithmic}
\If {$i \geq \text{maxval}$}
State $i \gets 0$
\Else
\If {$i+k \leq \text{maxval}$}
State $i \gets i+k$
\EndIf
\EndIf
\end{algorithmic}

\begin{verbatim}
if i \geq \text{maxval} then
  i \gets 0
else
  if i + k \leq \text{maxval} then
    i \gets i + k
  end if
end if
\end{verbatim}
Write Music
Write Music

It is possible to write music with LaTeX, so I decided to show some examples.

Symbols

\star \usepackage{wasysym}

\eightn \halfn \twon \fulln
\qn \natural \flat \sharp
Symbols

The \texttt{harmony} package offers some additional symbols:

\begin{verbatim}
\usepackage{harmony}

\AAcht ~~~ \Acht ~~~ \AchtBL ~~~ \AchtBR \\%
\DD ~~~ \DDohne ~~~ \Dohne ~~~ \Ds ~~~ \DS \%
\Ganz ~~~ \GaPa ~~~ \Halb ~~~ \HaPa ~~~ \Pu ~~~ \Sech \%
\SechBL ~~~ \SechBl ~~~ \SechBR ~~~ \SePa \%
\ViPa ~~~ \Zwdr ~~~ \ZwPa ~~~ \Vier ~~~ \AcPa
\end{verbatim}
### Tables

<table>
<thead>
<tr>
<th>Day</th>
<th>Min Temp</th>
<th>Max Temp</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>11°C</td>
<td>22°C</td>
<td>A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures.</td>
</tr>
<tr>
<td>Tuesday</td>
<td>9°C</td>
<td>19°C</td>
<td>Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest.</td>
</tr>
<tr>
<td>Wednesday</td>
<td>10°C</td>
<td>21°C</td>
<td>Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening.</td>
</tr>
</tbody>
</table>

#### Team sheet

<table>
<thead>
<tr>
<th>Role</th>
<th>GK</th>
<th>Paul Robinson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gk</td>
<td>LB</td>
<td>Lucus Radebe</td>
</tr>
<tr>
<td>Defenders</td>
<td>DC</td>
<td>Michael Duberry</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>Dominic Matteo</td>
</tr>
<tr>
<td></td>
<td>RB</td>
<td>Didier Domi</td>
</tr>
<tr>
<td>Midfielders</td>
<td>MC</td>
<td>David Batty</td>
</tr>
<tr>
<td></td>
<td>MC</td>
<td>Eirik Bakke</td>
</tr>
<tr>
<td></td>
<td>MC</td>
<td>Jody Morris</td>
</tr>
<tr>
<td>Forward</td>
<td>FW</td>
<td>Jamie McMaster</td>
</tr>
<tr>
<td>Strikers</td>
<td>ST</td>
<td>Alan Smith</td>
</tr>
<tr>
<td></td>
<td>ST</td>
<td>Mark Viduka</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team</th>
<th>P</th>
<th>W</th>
<th>D</th>
<th>L</th>
<th>F</th>
<th>A</th>
<th>Pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manchester United</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>Celtic</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Benfica</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>FC Copenhagen</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>
Tables

Tables are a common feature in academic writing, often used to summarise research results. Mastering the art of table construction in LaTeX is therefore necessary to produce quality papers and with sufficient practise one can print beautiful tables of any kind.

The `tabular` environment can be used to typeset tables with optional horizontal and vertical lines. LaTeX determines the width of the columns automatically.

The first line of the environment has the form:

\begin{tabular} [pos] {table spec}

\end{tabular}
Tables

The `tabular` argument tells LaTeX the alignment to be used in each column and the vertical lines to insert.

The number of columns does not need to be specified as it is inferred by looking at the number of arguments provided. It is also possible to add vertical lines between the columns here.
The following symbols are available to describe the table columns (some of them require that the package `array` has been loaded):

<table>
<thead>
<tr>
<th>l</th>
<th>left-justified column</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>centered column</td>
</tr>
<tr>
<td>r</td>
<td>right-justified column</td>
</tr>
<tr>
<td>p{'width'}</td>
<td>paragraph column with text vertically aligned at the top</td>
</tr>
<tr>
<td>m{'width'}</td>
<td>paragraph column with text vertically aligned in the middle (requires <code>array</code> package)</td>
</tr>
<tr>
<td>b{'width'}</td>
<td>paragraph column with text vertically aligned at the bottom (requires <code>array</code> package)</td>
</tr>
<tr>
<td></td>
<td>vertical line</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
By default, if the text in a column is too wide for the page, LaTeX won’t automatically wrap it. Using \( p\{ 'width' \} \) you can define a special type of column which will wrap-around the text as in a normal paragraph. You can pass the width using any unit supported by LaTeX, such as 'pt' and 'cm', or command lengths, such as \( \text{\textwidth} \).
Tables

The optional parameter `pos` can be used to specify the vertical position of the table relative to the baseline of the surrounding text. In most cases, you will not need this option. It becomes relevant only if your table is not in a paragraph of its own. You can use the following letters:

<table>
<thead>
<tr>
<th>pos</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>bottom</td>
</tr>
<tr>
<td>c</td>
<td>center (default)</td>
</tr>
<tr>
<td>t</td>
<td>Top</td>
</tr>
</tbody>
</table>
Tables

To specify a font format (such as bold, italic, etc.) for an entire column, you can add `\{\format\}` before you declare the alignment.

For example

\begin{tabular}{ >{\bfseries}l c >{\itshape}r }
will indicate a three column table with the first one aligned to the left and in bold font, the second one aligned in the center and with normal font, and the third aligned to the right and in italic.
Tables

The following symbols are available to describe the table columns (some of them require that the package `array` has been loaded):

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>column separator</td>
</tr>
<tr>
<td>\</td>
<td>start new row (additional space may be specified after \ using square brackets, such as [6pt])</td>
</tr>
<tr>
<td>\hline</td>
<td>horizontal line</td>
</tr>
<tr>
<td>\newline</td>
<td>start a new line within a cell (in a paragraph column)</td>
</tr>
<tr>
<td>\cline{i-j}</td>
<td>partial horizontal line beginning in column i and ending in column j</td>
</tr>
</tbody>
</table>

Note, any white space inserted between these commands is purely down to one's preferences. You can add spaces between to make it easier to read.
This example shows how to create a simple table in LaTeX. It is a three-by-three table, but without any lines.

\begin{tabular}{ l c r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
Expanding upon that by including some vertical lines:

\begin{tabular}{ l | c || r } 
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
Tables

To add horizontal lines to the very top and bottom edges of the table:

\begin{tabular}{ l | c || r }
\hline
1 & 2 & 3 \\ 
4 & 5 & 6 \\ 
7 & 8 & 9 \\ 
\hline
\end{tabular}
Tables

To add lines between all rows, as well as **centering** (notice the use of the **center** environment - of course, the result of this is not obvious from the preview on this slide):

\begin{center}
\begin{tabular}{ l | c || r }
\hline
1 & 2 & 3 \\
\hline
4 & 5 & 6 \\
\hline
7 & 8 & 9 \\
\hline
\end{tabular}
\end{center}
To add partial horizontal line beginning in column i and ending in column j:

\begin{tabular}{|r|l|}
  \hline
  7C0 & hexadecimal \\
  3700 & octal \\
  11111000000 & binary \\
  \hline
  1984 & decimal \\
  \hline
\end{tabular}
Text wrapping in tables

LaTeX's algorithms for formatting tables have a few shortcomings. One is that it will not automatically wrap text in cells, even if it overruns the width of the page.

For columns that will contain text whose length exceeds the column's width, it is recommended that you use the \texttt{p} attribute and specify the desired width of the column (although it may take some trial-and-error to get the result you want).

For a more convenient method, have a look at the \texttt{tabularx} package, or the \texttt{tabulary} package.

Instead of \texttt{p}, use the \texttt{m} attribute (requires array package) to have the lines aligned toward the middle of the box or the \texttt{b} attribute (requires array package) to align along the bottom of the box.
Text wrapping in tables - \texttt{p} attribute

Without specifying width for last column:

\begin{center}
\begin{tabular}{| l | l | l | l |}
\hline
Day & Min Temp & Max Temp & Summary
\hline
Monday & 11C & 22C & A clear day with lots of sunshine.
However, the strong breeze will bring down the temperatures.
\hline
Tuesday & 9C & 19C & Cloudy with rain, across many northern regions. Clear spells
across most of Scotland and Northern Ireland, but rain reaching the far northwest.
\hline
Wednesday & 10C & 21C & Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening.
\hline
\end{tabular}
\end{center}
Text wrapping in tables - p attribute

Without specifying width for last column, you get the following output:

<table>
<thead>
<tr>
<th>Day</th>
<th>Min Temp</th>
<th>Max Temp</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>11C</td>
<td>22C</td>
<td>A clear day with lots of sunshine. However, the strong breeze will...</td>
</tr>
<tr>
<td>Tuesday</td>
<td>9C</td>
<td>19C</td>
<td>Cloudy with rain, across many northern regions. Clear spells ac...</td>
</tr>
<tr>
<td>Wednesday</td>
<td>10C</td>
<td>21C</td>
<td>Rain will still linger for the morning. Conditions will improve b...</td>
</tr>
</tbody>
</table>

Note that the table has been cropped, since the output is wider than the page width.
Text wrapping in tables - \texttt{p} attribute

With width specified:

\begin{center}
\begin{tabular}{ | l | l | l | p{5cm} |}
\hline
Day & Min Temp & Max Temp & Summary \\
\hline
Monday & 11C & 22C & A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures. \\
\hline
Tuesday & 9C & 19C & Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest. \\
\hline
Wednesday & 10C & 21C & Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening. \\
\hline
\end{tabular}
\end{center}

The only difference is that the last column of the second one has a defined width of 5 centimeters, while in the first one we didn't specify any width.
Text wrapping in tables - `p` attribute

With width specified, you get the following output:

<table>
<thead>
<tr>
<th>Day</th>
<th>Min Temp</th>
<th>Max Temp</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>11C</td>
<td>22C</td>
<td>A clear day with lots of sunshine. However, the strong breeze will bring down the temperatures.</td>
</tr>
<tr>
<td>Tuesday</td>
<td>9C</td>
<td>19C</td>
<td>Cloudy with rain, across many northern regions. Clear spells across most of Scotland and Northern Ireland, but rain reaching the far northwest.</td>
</tr>
<tr>
<td>Wednesday</td>
<td>10C</td>
<td>21C</td>
<td>Rain will still linger for the morning. Conditions will improve by early afternoon and continue throughout the evening.</td>
</tr>
</tbody>
</table>

The only difference is that the last column of the second one has a defined width of 5 centimeters, while in the first one we didn't specify any width.
Defining multiple columns

It is possible to define many identical columns at once using the *{"num"}{"str"} syntax. This is particularly useful when your table has many columns. Here is a table with six centered columns flanked by a single column on each side:

\begin{tabular}{l*{6}{c}r}
Team & P & W & D & L & F & A & Pts \\
\hline
Manchester United & 6 & 4 & 0 & 2 & 10 & 5 & 12 \\
Celtic & 6 & 3 & 0 & 3 & 8 & 9 & 9 \\
Benfica & 6 & 2 & 1 & 3 & 7 & 8 & 7 \\
FC Copenhagen & 6 & 2 & 1 & 3 & 5 & 8 & 7 \\
\end{tabular}
Defining multiple columns

<table>
<thead>
<tr>
<th>Team</th>
<th>P</th>
<th>W</th>
<th>D</th>
<th>L</th>
<th>F</th>
<th>A</th>
<th>Pts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manchester United</td>
<td>6</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>Celtic</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Benfica</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>FC Copenhagen</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>

Here is a table with six centered columns flanked by a single column on each side:
Spanning

To complete this tutorial, we take a quick look at how to generate slightly more complex tables. Unsurprisingly, the commands necessary have to be embedded within the table data itself.

• Rows spanning multiple columns.

• Columns spanning multiple rows.
Rows spanning

The command for this looks like this:

\multicolumn{\textit{num_cols}}{\textit{alignment}}{\textit{contents}}

\texttt{num\_cols} is the number of subsequent columns to merge;

\texttt{alignment} is either \texttt{l, c, r,} or to have text wrapping specify a width \texttt{p{5.0cm}};

\texttt{contents} is simply the actual data you want to be contained within that cell.
Rows spanning

\begin{tabular}{ |l|l| }
\hline
\multicolumn{2}{|c|}{Team sheet} \hline
GK & Paul Robinson \hline
LB & Lucus Radebe \hline
DC & Michael Duberry \hline
DC & Dominic Matteo \hline
RB & Dider Domi \hline
MC & David Batty \hline
MC & Eirik Bakke \hline
MC & Jody Morris \hline
FW & Jamie McMaster \hline
ST & Alan Smith \hline
ST & Mark Viduka \hline
\end{tabular}
Columns spanning

The first thing you need to do is add `\usepackage{multirow}` to the preamble.

This then provides the command needed for spanning rows:

```
\multirow{''num_rows''}{''width''}{''contents''}
```

The arguments are pretty simple to deduce (* for the width means the content's natural width).
<table>
<thead>
<tr>
<th>Columns spanning</th>
</tr>
</thead>
</table>

```
\begin{tabular}{ |l|l|l| } \\
hline \\
\multicolumn{3}{ |c| }{Team sheet} \\
hline \\
\textbf{Goalkeeper} & \textbf{GK} & \textbf{Paul Robinson} \\
\hline \\
\textbf{Defenders} & LB & Lucus Radebe \\
& DC & Michael Duberry \\
& DC & Dominic Matteo \\
& RB & Didier Domi \\
\hline \\
\textbf{Midfielders} & MC & David Batty \\
& MC & Eirik Bakke \\
& MC & Jody Morris \\
\hline \\
\textbf{Strikers} & ST & Alan Smith \\
& ST & Mark Viduka \\
\hline \\
\end{tabular}
```

Columns spanning
Columns spanning

The main thing to note when using `\multirow` is that a blank entry must be inserted for each appropriate cell in each subsequent row to be spanned.

If there is no data for a cell, just don't type anything, but you still need the "&" separating it from the next column's data.

The astute reader will already have deduced that for a table of n columns, there must always be n-1 ampersands (&) in each row (unless `\multicolumn` is also used).
Floating table

• In WYSIWYG document processors, it is common to put tables in the middle of the text. This is what we have been doing until now.

• Professional documents, however, often make it a point to print tables on a dedicated page so that they do not disrupt the flow.

• From the point of view of the source code, one has no idea on which page the current text is going to lie, so it is hardly possible to guess which page may be appropriate for our table.
Floating table

• LaTeX can automate this task by abstracting objects such as tables, pictures, etc., and deciding for us where they might fit best.

• This abstraction is called a float. Generally, an object that is floated will appear in the vicinity of its introduction in the source file, but one can choose to control its position also.

• To tell LaTeX we want to use our table as a float, we need to place a tabular environment in a table environment, which is able to float and add a label and caption.
Floating table

• The table environment initiates a type of float just as the environment figure. In fact, the two bear a lot of similarities (positioning, captions, etc.).

• The environment names may now seem quite confusing. Let's sum it up:
  - tabular is for the content itself (columns, lines, etc.);
  - table is for the location of the table on the document, plus caption and label support.
Floating table

\begin{table}[position specifier]
  \centering
  \begin{tabular}{|l|}
    \hline
    ... your table ... \\
    \hline
  \end{tabular}
  \caption{This table shows some data}
  \label{tab:myfirsttable}
\end{table}
Floating table

In the table, we used a label, so now we can refer to it just like any other reference:

\ref{tab:myfirsttable}

The table environment is also useful when you want to have a list of tables at the beginning or end of your document with the command:

\listoftables

The captions show now up in the list of tables, if displayed.
Floating table

You can set the optional parameter `position specifier` to define the position of the table, where it should be placed. The following characters are all possible placements. Using sequences of it define your "wishlist" to LaTeX.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>where the table is declared (here)</td>
</tr>
<tr>
<td>t</td>
<td>at the top of the page</td>
</tr>
<tr>
<td>b</td>
<td>at the bottom of the page</td>
</tr>
<tr>
<td>p</td>
<td>on a dedicated page of floats</td>
</tr>
<tr>
<td>!</td>
<td>override the default float restrictions. E.g., the maximum size allowed of a \textbackslash b float is normally quite small; if you want a large one, you need this ! parameter as well.</td>
</tr>
</tbody>
</table>
Floating table

Default is \texttt{tbp}, which means that it is by default placed on the top of the page. If that's not possible, it's placed at the bottom if possible, or finally with other floating environments on an extra page.

You can force \LaTeX{} to use one given position. E.g. \texttt{[h]} forces \LaTeX{} to place it exactly where you place it (Except when it's really impossible, e.g. you place a table here and this place would be the last line on a page).

Again, understand it correctly: it urges \LaTeX{} to put the table at a specific place, but it will not be placed there if \LaTeX{} thinks it will not look great. If you really want to place your table manually, \texttt{do not use the table environment}.

Centering the table horizontally works like everything else, using the \texttt{\centering} command just after opening the \texttt{table} environment, or by enclosing it with a \texttt{center} environment.
Importing Graphics

Figure 1: A picture of a gull.

Figure 2: A picture of the same gull looking the other way!

Table 1: A simple table

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Incorporate how the tables and figures have independent counters.
Importing Graphics

There are two possibilities to include graphics in your document. Either create them with some special code, a topic which will be discussed in the next step, or import productions from third party tools, which is what we will be discussing here.

Strictly speaking, LaTeX cannot manage pictures directly: in order to introduce graphics within documents, LaTeX just creates a box with the same size as the image you want to include and embeds the picture, without any other processing. This means you will have to take care that the images you want to include are in the right format to be included.

This is not such a hard task because LaTeX supports the most common picture formats around.

Most drawing tools (e.g. for diagrams) can export in vector format. So you should always prefer PDF or EPS to PNG or JPG.
The \texttt{graphicx} package

As stated before, LaTeX can't manage pictures directly, so we will need some extra help: we have to load the \texttt{graphicx} package in the preamble of our document:

\begin{verbatim}
\usepackage{graphicx}
\end{verbatim}

This package accepts as an argument the external driver to be used to manage pictures; however, the latest version of this package takes care of everything by itself, changing the driver according to the compiler you are using, so you don't have to worry about this.

In many respects, importing your images into your document using LaTeX is fairly simple... once you have your images in the right format that is!
The `graphicx` package

As explained before, the image formats you can use depend on the driver that `graphicx` is using but, since the driver is automatically chosen according to the compiler, then the allowed image formats will depend on the compiler you are using.

Consider the following situation: you have added some pictures to your document in JPG and you have successfully compiled it in PDF. Now you want to compile it in DVI, you run latex and you get a lot of errors... because you forgot to provide the EPS versions of the pictures you want to insert.
The \texttt{graphicx} package

At the beginning of this presentation, we had stated that the same \LaTeX source can be compiled in both DVI and PDF without any change. This is true, as long as you don't use particular packages, and \texttt{graphicx} is one of those.

In any case, you can still use both compilers with documents with pictures as well, as long as you always remember to provide the pictures in two formats (EPS and one of JPG, PNG or PDF).
Including graphics

Now that we have seen which formats we can include and how we could manage those formats, it's time to learn how to include them in our document.

After you have loaded the `graphicx` package in your preamble, you can include images with `\includegraphics`, whose syntax is the following:

```
\includegraphics[attr1=val1, attr2=val2, ..., attrn=valn]{imagename}
```

As you should hopefully be aware by now, arguments in square brackets are optional, whereas arguments in curly braces are compulsory.
Including graphics

The argument in the curly braces is the name of the image. Write it without the extension. This way the LaTeX compiler will look for any supported image format in that directory and will take the best one (EPS if the output is DVI; JPEG, PNG or PDF if the output is PDF).

Images can be saved in multiple formats for different purposes. For example, a directory can have "diagram.pdf" for high-resolution printing, while "diagram.png" can be used for previewing on the monitor.
Including graphics

You can specify which image file is to be used by pdflatex through the preamble command:

\DeclareGraphicsExtensions{.pdf,.png,.jpg}

which specifies the files to include in the document (in order of preference), if files with the same basename exist, but with different extensions.
Including graphics

The variety of possible attributes that can be set is fairly large, so only the most common are covered below:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width = xx</td>
<td>Specify the preferred width of the imported image to xx.</td>
</tr>
<tr>
<td>height = xx</td>
<td>Specify the preferred height of the imported image to xx.</td>
</tr>
<tr>
<td>keepaspectratio</td>
<td>This can be set to either true or false. When true, it will scale the image according to both height and width, but will not distort the image.</td>
</tr>
<tr>
<td>scale = xx</td>
<td>Scales the image by the desired scale factor. e.g., 0.5 to reduce by half, or 2 to double.</td>
</tr>
<tr>
<td>angle = xx</td>
<td>This option can rotate the image by xx degrees (counter-clockwise).</td>
</tr>
<tr>
<td>trim = l b r t</td>
<td>This option will crop the imported image by l from the left, b from the bottom, r from the right, and t from the top. Where l, b, r and t are lengths.</td>
</tr>
<tr>
<td>clip</td>
<td>For the trim option to work, you must set clip = true.</td>
</tr>
<tr>
<td>page = x</td>
<td>If the image file is a pdf file with multiple pages, this parameter allows you to use a different page than the first.</td>
</tr>
<tr>
<td>resolution = x</td>
<td>Specify image resolution in dpi.</td>
</tr>
</tbody>
</table>
Including graphics

In order to use more than one option at a time, simply separate each with a comma.

The order you give the options matters. E.g you should first rotate your graphic (with angle) and then specify its width.

Included graphics will be inserted just there, where you placed the code, and the compiler will handle them as "big boxes".

As we will see in the floats section, this can disrupt the layout; you'll probably want to place graphics inside floating objects.
Example

OK, it's time to see graphicx in action. Say you had a file 'chick.PNG' you would include it like:

\includegraphics{chick}

This simply imports the image, without any other processing. However, it is very large/small (so we won't give an example of how it would look here!)
Example

So, let's scale it down:

\includegraphics[scale=0.5]{chick}

This has now scaled it by half.
Example

If you wish to be more specific and give actual lengths of the image dimensions, this is how to go about it:

\includegraphics[width=3.5cm]{chick}
Example

To rotate (I also scaled the image down):

\includegraphics[angle=180, scale=0.5]{chick}
Example

And finally, an example of how to crop an image should you wish to focus on one particular area of interest:

%trim option's parameter order: left bottom right top

\includegraphics[trim = 10mm 80mm 20mm 5mm, clip, width=3cm]{chick}
Graphics storage

The command \texttt{\textbackslash graphicspath} tells \LaTeX{} \texttt{where to look for images}, which can be useful if you store images centrally for use in many different documents.

The \texttt{\textbackslash graphicspath} command takes one argument, which specifies the additional paths you want to be searched when the \texttt{\textbackslash includegraphics} command is used.
Graphics storage

Here are some examples:

\graphicspath{ {/var/lib/images/} }

\graphicspath{ {./images/} }

Using absolute paths, \texttt{\graphicspath} makes your file less portable, while using relative paths (like the second example), there should not be any problem with portability.
Floating figure

LaTeX by default recognizes "table" and "figure" floats.

Floats are there to deal with the problem of the object that won't fit on the present page, and to help when you really don't want the object here just now.

Floats are not part of the normal stream of text, but separate entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the designer specifies).
Floating figure

They always have a caption describing them and they are always numbered so they can be referred to from elsewhere in the text.

If there is not enough room on the current page, the float is moved to the top of the next page.

This can be changed by moving the Table or Figure definition to an earlier or later point in the text, or by adjusting some of the parameters which control automatic floating.
Floating figure

To create a figure that floats, use the \texttt{figure} environment.

\begin{figure}[placement specifier]
    \centering
    ... figure contents ...
    \caption{This figure shows some data}
    \label{fig:myfirstfigure}
\end{figure}
**Floating figure**

In the figure, we used a label, so now we can refer to it just like any other reference:

\ref{fig:myfirstfigure}

The *figure* environment is also useful when *you want to have a list of figures* at the beginning or end of your document with the command:

\listoffigures

The captions show now up in the list of figures, if displayed.
Floating figure

You can set the optional parameter placement specifier to define the position of the figure, where it should be placed, and its purpose is to give the author a greater degree of control over where certain floats are placed.

<table>
<thead>
<tr>
<th>h</th>
<th>place the float here, i.e., approximately at the same point it occurs in the source text (however, not exactly at the spot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>position at the top of the page</td>
</tr>
<tr>
<td>b</td>
<td>position at the bottom of the page</td>
</tr>
<tr>
<td>p</td>
<td>put on a special page for floats only</td>
</tr>
<tr>
<td>!</td>
<td>override internal parameters LaTeX uses for determining &quot;good&quot; float positions</td>
</tr>
</tbody>
</table>
Example

The location of the caption is traditionally underneath the float.

\begin{figure}[h!]
  \centering
  \includegraphics[width=5cm]{Gull}
  \caption{A picture of a gull.}
  \label{fig:gull}
\end{figure}

Gaviota Dominicana - Larus Dominicanus (1.1).
Example

a trick ... you can look for others ... good job!

\begin{figure}[h!]
  \centering
  \reflectbox{
    \includegraphics[width=5cm]{Gull}}
  \caption{A picture of the same gull looking the other way!}
  \label{fig:gull2}
\end{figure}

Gaviota Dominicana - Larus Dominicanus (\ref{fig:gull1}) and the same gull looking the other way (\ref{fig:gull2}).
PGF/TikZ

One possible solution how to draw graphics directly with TeX commands is PGF/TikZ.

TikZ can produce portable graphics in both PDF and PostScript formats using either plain (pdf)TEX, (pdf)Latex.

It comes with very good documentation and an extensive collection of examples: www.texample.net/tikz/

PGF ("portable graphics format") is the basic layer, providing a set of basic commands for producing graphics, and TikZ is the frontend layer with a special syntax, making the use of PGF easier.
Using TikZ in a LaTeX document requires loading the tikz package, somewhere in the preamble:

\usepackage{tikz}

This will automatically load the pgf package. To load further libraries use:

\usetikzlibrary{list of libraries separated by commas}

Examples for libraries are "arrows", "shapes.geometric", "shapes.miss", ecc.
Drawing commands have to be enclosed in an `tikzpicture` environment:

\begin{tikzpicture}[options]
  ... tikz commands ...
\end{tikzpicture}

or alternatively:

\tikz[options]{tikz commands}

A path is a series of straight and curved line segments. **The instruction has to end with a semicolon (;).**

\path[options] specification;
Example

We can create graphic elements by defining some of their key properties.

\begin{tikzpicture}
\draw[gray, thick] (-1,2) -- (2,-4);
\draw[gray, thick] (-1,-1) -- (2,2);
\filldraw[black] (0,0) circle (2pt) node[anchor=west] {Intersection point};
\end{tikzpicture}
Example

These elements can be combined to create more elaborated figures.

\begin{tikzpicture}
\draw (-2,0) -- (2,0);
\filldraw [black] (0,0) circle (2pt);
\draw (-2,-2) .. controls (0,0) .. (2,-2);
\draw (-2,2) .. controls (-1,0) and (1,0) .. (2,2);
\end{tikzpicture}
Curriculum Vitae
Curriculum Vitae

A curriculum vitae or résumé has a universal requirement: its formatting must be flawless.

This is a great example of cases where the power of LaTeX comes to the front.

Thanks to its strong typographical stance, LaTeX is definitely a document processor of choice to write a CV.

Of course you can design your own CV by hand. Otherwise, you may want to use a dedicated class for that task. This way, writing a CV in LaTeX is as simple as filling the forms, and you are done.
As of 11 March 2002 the European Commission has defined a common format for curricula vitae.

This class is an unofficial LATEX implementation of the standard model for curricula vitae (the Europass CV) as recommended by the European Commission.

The Europass CV defines both the content and the layout of a curriculum vitae.

The europecv class provides support for the latter.
http://www.ctan.org/pkg/europecv
europecv IT EN

\documentclass[totpages,helvetica,openbib,italian]{europecv}
\usepackage[T1]{fontenc}
\usepackage{graphicx}
\usepackage[a4paper,top=1.27cm,left=1cm,right=1cm,bottom=2cm]{geometry}
\usepackage[italian]{babel}
\usepackage{bibentry}
\usepackage{url}

.....
.....
.....
.....
.....
Informazioni personali

- Cognome/i Nome/i
- Indirizzo/i
- Telefono/i
- Fax
- Email
- Nazionalità
- Data di nascita
- Sesso

Impiego ricercato/
Settore di competenza

Esperienza professionale

Cognome/i Nome/i
- Numero civico, via, codice postale, città, nazione
- Facoltativo  Mobile: Facoltativo
- Facoltativo

- email@address.com Facoltativo
- Facoltativo
- Facoltativo
- Facoltativo

Facoltativo
moderncv

A modern curriculum vitae class.

The class provides facilities for typesetting modern curricula, both in a classic and in a casual style.

It is fairly customizable, allowing you to define your own style by changing the colours, the fonts, etc.

http://www.ctan.org/pkg/moderncv
moderncv

%% start of file `template.tex'.
%% Copyright 2006-2013 Xavier Danaux (xdanaux@gmail.com).
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License version 1.3c,
% available at http://www.latex-project.org/lppl/.

\documentclass[11pt,a4paper,sans]{moderncv}

....

....
moderncv - casual - blue -

John Doe

Some quote

Education
- year-year Degree, Institution, City, Grade.
- Description
- year-year Degree, Institution, City, Grade.
- Description

Master thesis
- title Title
- Supervisors
- Short thesis abstract

Experience
- Vocational
- year-year Job title, Employer, City.
Curriculum Vitae

moderncv - classic - green -

John Doe

ResumĂł title

Some quote

Education

year-year Degree, Institution, City, Grade.
Description

year-year Degree, Institution, City, Grade.
Description

Master thesis

title Title
supervisors Supervisors
description Short thesis abstract

Experience

Vocational

year-year Job title, Employer, City.
General description no longer than 1–2 lines.
more information

![Google Logo](https://www.google.com)

latex

Cerca con Google  Mi sento fortunato

... syntax ... ideas ... examples ...

... and much more !!
Thanks for your attention