LaTeX
A
Beginner's
Guide

Sayak Haldar
sayakhaldar@ymail.com
Shuvam Bosana
shuvambosana0705@gmail.com
Subham Banerjee
subhambansphs@gmail.com
Student of
Computer Science & Technology
Bengal Engineering & Science University,Shibpur.

Abstract

Suppose, at the time of the submission of your project, your project mentor tells you to submit the corresponding documentation in LaTeX . Obviously, that does not require your expertisation in LaTeX . You just need to learn the survival tricks in that case. Well, this document will provide you the survival tricks for such conditions.

Contents

1 Different documentclasses in LaTeX :

- **Article:** For articles in scientific journals, presentations, short reports, program documentation, invitations, ...
- **IEEEtran:** For articles with the IEEE Transactions format.
- **proc:** A class for proceedings based on the article class.
- **minimal:** Is as small as it can get. It only sets a page size and a base font. It is mainly used for debugging purposes.
- **report** For longer reports containing several chapters, small books, thesis, ...
- **book:** For writing books.
- **slides:** For slides. The class uses big sans serif letters.
- **Memoir:** For changing sensibly the output of the document. It is based on the book class, but you can create any kind of document with it.
- **Letter:** For writing letters.
- **Beamer:** For creating presentations.

There are some additional packages too like beamerposter. Beamerposter is used for creating educational and other types of poster.

2 Document class options:

10pt, 11pt, 12pt	Sets the size of the main font in the document. If no option is specified, 10pt is assumed.
a4paper, letterpaper,...	Defines the paper size. The default size is letterpaper; However, many European distributions of TeX now come pre-set for A4, not Letter, and this is also true of all distributions of pdfLaTeX. Besides that, a5paper, b5paper, executivepaper, and legalpaper can be specified.
fleqn	Typesets displayed formulas left-aligned instead of centered
continued to next page	

	continued from previous page
leqno	Places the numbering of formulas on the left hand side instead of the right.
titlepage, notitlepage	Specifies whether a new page should be started after the document title or not. The article class does not start a new page by default, while report and book do.
twocolumn	Instructs LaTeX to typeset the document in two columns instead of one.
twoside, oneside	Specifies whether double or single sided output should be generated. The classes article and report are single sided and the book class is double sided by default. Note that this option concerns the style of the document only. The option twoside does not tell the printer you use that it should actually make a two-sided printout.
landscape	Changes the layout of the document to print in landscape mode.
openright, openany	Makes chapters begin either only on right hand pages or on the next page available. This does not work with the article class, as it does not know about chapters. The report class by default starts chapters on the next page available and the book class starts them on right hand pages.
draft	makes LaTeX indicate hyphenation and justification problems with a small square in the right-hand margin of the problem line so they can be located quickly by a human. It also suppresses the inclusion of images and shows only a frame where they would normally occur.

Now, we are going to discuss the basics of creating an article using LaTeX .

3 Creating an article using LaTeX :

First, you have to enter the following command in the preamble of your document:

```
\documentclass{article}
```

This is to be mentioned, because When processing an input file, LaTeX needs to know the type of document the author wants to create. This is specified with that command & It is recommended to put this declaration at the very beginning.

Another thing is that, for every type of document, you have to use **\begin{document}** at the beginning and **\end{document}** at the end. The meaning of these two commands are given in the folllowing table:

\begin{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.
\end{document}	The document environment ends here. It tells LaTeX that the document source is complete, anything after this line will be ignored.

3.1 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 will produce output like: It does not matter whether you enter one or several spaces after a word.

An empty line starts a new paragraph.
As you can see, you cannot give desired horizontal spaces by just pressing 'space' or 'tab' for multiple time. You, might also notice that pressing 'enter' for several times do not produce the desired vertical space. Here, we are going to tell you the appropiate way of doing it.
For giving horizontal spaces, you have to use :

```
\hspace{length}
```

Or,

```
\hspace*{length} where length will be provided in the length
units used in latex.
```

We will discuss about the existing LaTeX length units later.

Now, we will discuss the difference between **\hspace*{length}** and **\hspace{length}**:

By using **\hspace{length}** we could provide horizontal spaces between two words. but if we want to provide horizontal space at the beginning of a sentence we have to use **\hspace*{length}**.So, considering all situations it is better to use **\hspace*{length}** all the time.
And for producing the desired vertical spaces, you have to use:

```
\vspace{length}
```

Or,

```
\vspace*{length}
```

The difference between **\vspace{length}** and **\vspace*{length}**: **\vspace{length}** should normally be used between two lines. But if the space is needed at the top of a page or at the bottom of a page, then **\vspace*{length}** should be used. Additional space between two lines of the same paragraph or within a table is specified by the following command:

\\[**Length**] where length should be provided with proper magnitude and unit.

3.2 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.

```
# $ % ^ & _ { } ~ \
```

As you will see, these characters can be used in your documents all the same by adding a prefix backslash:
\# \$ \% \^{} \& _ \{ \} \~{} \textbackslash{}
Also, you cannot typeset '>' and '<' directly form keyboard. While you are in text mode(Later, you will come to know that those signs can be produced in math mode by using different commands) you have to use **'\textgreater'** and **'\textless'** for producing '>' and '<' respectively.

3.3 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 example%of using comment.
```

It will produce output like:
This is an example
As you can see, 'of using comment' portion does not show up in the printed version.
Now, we are going to discuss about top matter of a document.

3.4 Top matter of a document:

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 these types of informations within LaTeX is collectively referred to as top matter.
A simple example:

```
\begin{document}
\title{LaTeX-A Beginner's Guide}
\author{Sayak Haldar}
\date{December,2013}
\maketitle
\end{document}
```

The \title, \author, and \date commands are self-explanatory. You have to put the title, author name, and date in curly braces after the relevant command. The title and author are usually compulsory (Those things are not written by LateX automatically);if you want to omit the \date command, LaTeX uses today's date by default. You have to finish the top matter with the **\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. If you omit \maketitle, the titling will never be typeset (unless you write of your own).

A more complicated example:

```
\title{\LaTeX\-A Beginner's Guide}
\author{Sayak Haldar\\
Bengal Engineering \& Science University,\\
Shibpur\\
\texttt{sayakhaldar@ymail.com}}
\date{\today}
\maketitle
```

Now, suppose you have multiple authors for a particular document, so you need to add mulitple authors' name as well as their email addresses in the author field. We will provide an example of it.

```
\title{\LaTeX\-A Beginner's Guide}
\author{Sayak Haldar(111205038)\\
\texttt{sayakhaldar@ymail.com}
\and
Shuvam Bosana(111205042)\\
\texttt{shuvambosana0705@gmail.com}
\and
Subham Banerjee(111205014)\\
```

continued to next page

continued from previous page

```
\texttt{subhambansphs@gmail.com}
}
\date{today}
\maketitle
```

Note: However,the command **\maketitle** is to create a seperate title page, if you dont want a seperate title page and want to write the abstract of the document just after the mentioned date you could omit the command **\maketitle**. And if you are planning to typeset a titlepage by copying one of the given codes, also do not forget to mention the documentclass at the very beginning of your LaTeX code.

3.5 Adding abstract to a document:

An abstract is a brief summary of a research article, thesis, review, conference proceeding or any in-depth analysis of a particular subject or discipline, and is often used to help the reader quickly ascertain the paper's purpose. So, adding abstract to a document is very important. It is to be written at the start of a document. This could be done by:

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

The output would be like:

Abstract

Your abstract here.

Though an abstract is only added at the beginning of a document or at the beginning of a chapter(If a book is written), we just want you to show the output.

3.6 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 found in simple.tex.

```
\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...
```

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 table below). Each section in this table is a subsection of the one above it.

Command	Level	comments
\part{"part"}	-1	not in letters
\chapter{"chapter"}	0	only books and reports
\section{"section"}	1	not in letters
\subsection{"subsection"}	2	not in letters
\subsubsection{"subsubsection"}	3	not in letters
\paragraph{"paragraph"}	4	not in letters
\subparagraph{"subparagraph"}	5	not in letters

All the titles of the sections are added automatically to the table of contents (if you decide to insert one). 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:

Suppose, your document has a very long section title :'An analysis of the effect of the revised recruitment policies on staff turnover at divisional headquarters'. Now, if you create a table of contents automatically(using the **\tableofcontents** command this long section title can cause you troubles. So, if you want to show a small section title in the Table of Contents for this particular section, say 'Effect on staff turnover' you have to use the following command:

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

Instead of

```
\section{An analysis of the effect of the revised
recruitment policies on staff turnover at divisional
headquarters}
```

3.6.1 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.).

You can change the depth to which section numbering occurs, so you can turn it off selectively. By default it is set to 2. If you only want parts, chapters, and sections numbered, not subsections or subsubsections etc., you can change the value of the **secnumdepth** counter using the **\setcounter** command, giving the depth level you wish. For example, if you want to change it to "1":

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

A related counter is **tocdepth**, which specifies what depth to take the Table of Contents to. It can be reset in exactly the same way as **secnumdepth**. For example:

```
\setcounter{tocdepth}{3}
```

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}
```

All the divisional commands from **\part*** to **\subparagraph*** have this "starred" version which can be used on special occasions for an unnumbered heading when the setting of secnumdepth would normally mean it would be numbered.

To my opinion, unnumbered section,subsections look better than numbered sections,subsections etc. But then creating 'Table of Contents' will be a problem. You will not get a 'Table of Contents' by just using **\tableofcontents** then. You have to enlist all the sections,subsections etc... manually by following the given command pattern:

```
\section*{Introduction}
\addcontentsline{toc}{section}{Introduction}
```

3.7 Table of contents in an article:

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 pagenumber references are correctly calculated. We've already seen how to use the optional argument to the sectioning commands to add text to the ToC which is slightly different from the one printed in the body of the document. And if you create your whole document using unnumbered section then it is also possible to force them in the 'Table of Contents' using the following command pattern:

```
\section*{Introduction}
\addcontentsline{toc}{section}{Introduction}
```

3.7.1 Depth of Table of Contents:

The default ToC will list headings of level 3 and above. To change how deep the table of contents displays automatically the following command can be used in the preamble:

```
\setcounter{tocdepth}{4}
```

This will make the table of contents include everything down to paragraphs. The levels are defined above on this page. Note that this solution does not permit changing the depth dynamically.

3.8 Fontsize:

3.8.1 customization of the fontsize for the whole document:

Since most the documentclasses of LaTeX provides 10pt-12pt as default fontsize(excluding memoir documentclass). However, we must need other ways to use fontsizes other than 10pt,11pt or 12pt.
A simple solution to this problem is to use the extsizes bundle. This bundle offers "extended" versions of the article, report, book and letter classes, at sizes of 8, 9, 14, 17 and 20pt as well as the standard 10–12pt.

Now, one question is obvious, how to use extsize? Now, suppose, you are creating an article. Then the command would be like:

```
\documentclass[fontsize]{extarticle}
where fontsize can be 8,9.10,11,12,14,17 and 20 pt.
```

But this provides you only a few options which may not be optimal. More satisfactory are the KOMA-script classes, which are designed to work properly with the class option files that come with extsizes, and the memoir class that has its own options for document font sizes 9pt–12pt, 14pt, 17pt, 20pt, 25pt, 30pt, 36pt, 48pt and 60pt. The classes also offer size setup for any old font size, and the scrextend package can extend this facility for use with any class. As an example:

```
\usepackage[fontsize=12.3]{scrextend}
```

It will indeed set up the main document font to have size 12.3pt with an appropriate default baselineskip.

3.8.2 customization of the fontsize for a particular portion of the text

You can change the fontsize of a paricular portion of the text by using the following options:

```
{\Huge{This text is huge}}

{\Large{This text is large}}

{\normalsize{This text is normalsize}}

{\small{This text is small}}

{\footnotesize{This text is footnotesize}}

{\scriptsize{This text is scriptsize}}

{\tiny{This text is tiny}}
```

these commands will produce like:

{\Huge This text is huge}

{\Large This text is large}

{\normalsize This text is normalsize}

{\small This text is small}

{\footnotesize This text is footnotesize}

{\scriptsize This text is scriptsize}

{\tiny This Text is tiny}

But note that, these commands will only change the fontsize of a particular text portion, but these will not affect the the fontsize of the whole document.

Now, by using these commands, you used the already defined LaTeX commands for fontsize. Later, you will learn to define your own fontsize.

3.9 Papersize:

If no papersize is specified, then the papersize is set to letterpaper.This is shorter by 18 mm (about 3/4 inch), and slightly wider by 8 mm (about 1/4 inch), compared to A4 (which is the standard in almost all the rest of the world).Besides that, a5paper, b5paper, executivepaper, and legalpaper can be specified.

It can be mentioned by the following command:

```
\documentclass[a4paper]{article}
```

However, we can explore more papersize options with geometry package. The immediate advantage of this package is that it lets you customize the page size even with classes that do not support the options. For instance, to set the page size, add the following to your preamble:

```
\usepackage[a4paper]{geometry}
```

The geometry package has many pre defined page sizes. like-
a0paper,a1paper,. . . a6paper
b0paper,b1paper,. . . b6paper
legalpaper
legalpaper
executivepaper

To explicitly change the paper dimensions using the geometry package, the paperwidth and paperheight options can be used. For example:

```
\usepackage[paperwidth=5.5in, paperheight=8.5in]{geometry}
```

Otherwise we could change the length manually by using the **\setlength** command.

You have to define **\paperwidth** and **\paperheight** in the preamble of your document in all cases.

And,

After the preamble, **\pdfpagewidth** and **\pdfpageheight** if you are using pdftex.
we will discuss about different types of lengths of LaTeX later.

3.10 Language:

If you did not specify a particular language, then the language is set to english.Though it can be specified in the option section of the following command in the preamble:

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

However, LaTeX can be configured and used appropiately to write documents other than english. But one must concentrate on three main points before creating a document in LaTeX in other languages.

- LaTeX needs to know how to hyphenate the language(s) to be used.
- The user needs to use language-specific typographic rules. In French for example, there is a mandatory space before each colon character (:).
- The input of special characters, especially for languages using an input system (Arab, Chinese, Japanese, Korean).

To use language other than english we have to use a package named 'Babel'. This is to be done by using the following command in the preamble section.

```
\usepackage[language]{babel}
```

But You should place it soon after the \documentclass command, so that all the other packages you load afterwards will know the language you are using. Babel will automatically activate the appropriate hyphenation rules for the language you choose. If your LaTeX format does not support hyphenation in the language of your choice, babel will still work but will disable hyphenation, which has quite a negative effect on the appearance of the typeset document. Babel also specifies new commands for some languages, which simplify the input of special characters.

3.10.1 Other language supported by babel:

The languages supported by babel together with the name of options with which you can load babel for each language,are:-

Afrikaans	afrikaans
Bahasa	bahasa
Breton	breton
Catalan	catalan
Croatian	croatian
Czech	czech
Danish	danish
Dutch	dutch
English	english, USenglish, american, UKenglish, british
Esperanto	esperanto
Estonian	estonian
Finnish	finnish
French	french, francais
Galician	galician
German	austrian, german, germanb
Greek	greek
Hebrew	hebrew
Hungarian	magyar, hungarian
Irish Gaelic	irish
Italian	italian
Lower Sorbian	lowersorbian
Norwegian	norsk, nynorsk
Polish	polish
Portuguese	portuges, portuguese, brazilian, brazil
Romanian	romanian
Russian	russian
Scottish Gaelic	scottish
Spanish	spanish
Slovakian	slovak
Slovenian	slovene
Swedish	swedish
Turkish	turkish
Upper Sorbian	uppersorbian
Welsh	welsh

3.10.2 Use of some basic commands provided by babel:

If you call babel with multiple language:

```
\usepackage[languageA,languageB]{babel}
```

then the last language in the option list will be active (i.e. languageB).

An example:-Suppose you are writing an article in english which contains some french phrase then in the preamble section after declaring the documentclass you should use the command:

```
\usepackage[english,french]{babel}
```

Now, by default the languageB i.e the french language is active. But as the examle says, your article only contains a few french phrases. So, most of the time you need languageA i.e english language to be active. Now to set english language to be active, you need to use the command:

```
\selectlanguage{english}
```

Now, to write the french phrases you need to use the following command whenever you need it:

```
\foreignlanguage{french}{French phrase}
```

But suppose, in another example you use the same languages i.e english and french and active language in english at some instant of time and you need to write a whole paragraph in french. Then, what will you do?
babel provides a simple environment for entering larger pieces of text in another language:

```
\begin{otherlanguage}{french}
paragraph in french language.
\end{otherlanguage}
```

our example ends here.

3.11 Basic font formatting:

3.11.1 a particular text potion in bold font:

There are three ways to do it:

```
1.{\textbf{Some text in bold}}
```

this will give output like:
Some text in bold
or,

```
2.{\bfseries{Some text in bold}}
```

This will also provide the same output.
And the last way to make a text bold is:-

```
3.{\bf{Some text in bold}}
```

Note:-you must enclose the **\textbf** or **\bfseries** or **\bf** command as well as the text portion by curly bracket properly. Otherwise the whole text will become bold from the portion from where we use the command **\textbf** or **\bfseries** or **\bf**

3.11.2 A particular text portion in italics font:

First way to do it is to use **\textit** command.
An example:

```
{\textit{Some text in italics}}
```

This will produce output like:
Some text in italics Another way of doing it is to use **\itshape** command.

```
{\itshape{Some text in italics}}
```

3.11.3 A particular text portion emphasized:

The command to emphasize a particular line or word is:

```
\emph{Emphasized text}
```

This will give output like:
Emphasized text

Note:-Emphasizing a text will provide an output which is quite similar to make a text in italics. But theses two are different things.

3.11.4 Using underline under a text portion:

```
\underline{Underlined text}
```

This will provide output like:
Underlined text

3.12 Line Breaking and Page Breaking:

\newline: Breaks the line at the point of command.

\\: Breaks the line at the point of command.It's a shorter version of the previous command but it does exactly the same.

***:** Breaks the line at the point of command & additionally prohibits a page break after the forced line break.

\linebreak[number]	:Breaks the line at the point of command. The number you provide as an argument represents the priority of the command in a range from 0(it will be easily ignored) to 4(do it anyway). LaTeX will try to produce the best line breaks possible, meeting it's high standard. If it cannot, it will decide whether including the line break or not according to the priority you have provided.
\newpage	:Ends the current page and starts a new one.
\pagebreak{number} :	Breaks the current page at the point of command.
\nopagebreak{number}:	the page being broken at the point of command. The optional number argument sets the priority at a scle from 0 to 4.
\clearpage:	End the current page and causes any floats encountered in the input,but yet to appear, to be printed.

3.13 Bullets and numberings:

3.13.1 itemize:

This environment is for your standard bulleted list of items.
An example:

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

The code above will produce output like:

- The first item

- The second item
- The third etc ...

3.13.2 enumerate:

The enumerate environment is for ordered lists, where by default, each item is numbered sequentially.
An example:

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

This will provide output like:

1. The first item
2. The second item
3. The third etc ...

enumerate counter is by default set to 0. But it can be manually set to any no. For instance if you set the counter as 4, The first no. which will come up is 5. Now, the counter is set by the following command(if you are not using nested loop):

```
\setcounter{enumi}{number}
```

Now, suppose you want to set the counter to 4, then the first number which will come up is 5. This example is illustrated below:

5. fifth element

And this is done by the code:

```
\begin{enumerate}
  \setcounter{enumi}{4}
  \item fifth element
\end{enumerate}
```

Now, the bullets and numberings have a depth of 4(by default. If you need more than four, you have to use easylist package). Now, if you want to use nested loop of numberings and want to set the counter for each of the four levels manually then, you have to use the commands:

```
\labelenumi
\labelenumii
\labelenumiii
\labelenumiv
```

3.13.3 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.
An example:

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

This will provide output like:

First The first item

Second The second item

Third The third etc ...

3.13.4 Nested list:

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 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.
An example:

```
\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}
```

This will provide output like:

1. The first item
   (a) Nested item 1
   (b) Nested item 2
2. The second item
3. The third etc ...

3.13.5 Customization of bullets:

Suppose, you want to change the bullet chape to rightarraw. Then you have to use:

```
\begin{itemize}
\renewcommand{\labelitemi}{$\Rightarrow$}
\item The first item.
\item The second item.
\item the third item etc\ldots
\end{itemize}
```

This will provide output like:

⇒ The first item.

⇒ The second item.

⇒ the third item etc...

You could use other characters as bullets. like:

- ∘ An open circle as a bullet.
- · A centered dot as a bullet.
- ⋆ A five pointed star as a bullet.
- ∗ A centered asterisk as a bullet.
- ♢ An open diamond as a bullet.

The change of shape in bullet is done by the command:

```
\renewcommand{\labelitemi}{$\bullet$} where:
\bullet=\circ,\cdot,\star, \ast,\diamondsuit
for providing open circle, centered dot,five pointed
star,centered asterisk,diamondsuit as bullets respectively.
```

3.14 Page numbering:

In LaTeX , page numbering starts from 1 by default. If you want to omit page number from a particular page you need to use the command:

```
\thispagestyle{empty}
```

This command should be used at the beginning of that page. i.e, after using **\newpage** or **\clearpage** command.
That will remove the page number from that particular page. After omitting page number from a particular page you have to set the page numbering for the next page.We will illustrate the reason of it by giving an example.
Suppose, you want to omit the page number from the third page. If you dont set the page numbering for the next page, the page number of next page will be four. But, this is undesirable. So you have to use the following command at the beginning of fourth page:

```
\setcounter{page}{3}
```

But again, this could be only be used after **\newpage** or **\clearpage** command.

3.15 Colors:

By default, the textcolor is black and the background color is white while we are creating a document in LaTeX .

But, if we use color package we can use more colors as textcolor and background-color option.

3.15.1 Useful packages:

To make use of these color features either the color package or the xcolor package must be inserted into the preamble.

```
\usepackage{color}
```

This is to include the color package.

```
\usepackage[usenames,dvipsnames,svgnames,table]{xcolor}
```

This is to include the xcolor package.

The **\usepackage** is obvious, but the initialization of additional commands like usenames allows you to use names of the default colors, the same 16 base colors as used in HTML. The dvipsnames allows you access to more colors, another 64, and svgnames allows access to about 150 colors. The initialization of "table" allows colors to be added to tables by placing the color command just before the table. The package loaded here is the xcolor package.

Note: The sixteen base colors of HTML are black,gray,silver,white,maroon,red,olive,yellow,green,lime,teal,aqua, navy,blue,purple,fuchsia. For knowing more details, the reader is requested to google about it.

Similarly, the reader is requested to google about the colors provided by svg-names from the xcolor package if he/she wants to know more about it.

3.15.2 Entering color text:

The simplest way to type colored text is by:
\textcolor{declared-color}{text}

where declared-color is a color that was defined before by \definecolor or we can use predefined colors. Another possible way to use different textcolors is: {\color{declared-color} some text}

The difference between \color{} and \textcolor{}:
The \color environment allows the text to run over multiple lines and other text environments whereas the text in \textcolor must all be one paragraph and not contain other environments.

3.15.3 Changing the background color of a whole page:

You can change the background color of the whole page by:

```
\pagecolor{declared-color}
```

3.15.4 Using color box:

If You just want to change the background color of the text:

```
\colorbox{declared-color}{text}
```

Suppose you want the background color of a particular line as gray,then you have to use:

```
\colorbox{gray}{Here, the backgroundcolor is gray.}
```

Thsi will produce output like:

Here, the backgroundcolor is gray

If you want both the background color as well as textcolor to be changed,you have to use:

```
\colorbox{declared-color1}{\color{declared-color2}text}
```

Suppose, we want the textcolor as white and the backgroundcolor as gray for a particular line,then we have to use:

```
\colorbox{gray}{\color{white}Here, the textcolor
is white and backgroundcolor is gray}
```

And it will produce output like:

Here, the textcolor is white and backgroundcolor is gray

There is also **\fcolorbox** to make framed background color in yet another color:

`\fcolorbox{declared-color-frame}{declared-color-background}{text}`

Now, a complicated example: Suppose, you want your framecolor as black,background as yellow and textcolor as black,then you have to use the command:

```
\fcolorbox{black}{yellow}{\color{black}Using fcolorbox}
```

And the output would be like:

Using fcolorbox

This color combination isn't that good.Is it?
Anyway, you could try some different color combinations.

3.15.5 Predefined colors:

The predefined color names are
white, black, red, green, blue, cyan, magenta, yellow.
There may be other pre-defined colors on some other system, but these should be available on all systems.
We can even use one of the 68 dvips colors, or define our own. These options will be discussed later.

3.16 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 Creating Graphics part, (see Introducing Procedural Graphics) or import productions from third party tools, which is what we will be discussing here.
But 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.
Now, as we stated before, LaTeX can't manage pictures directly, so you will need some extra help: you have to load the graphicx package in the preamble of your document:

```
\usepackage{graphicx}
```

3.16.1 document options:

The graphics and graphicx packages recognize the draft and final options given in the \documentclass[...]{...} command at the start of the file. (See Document Classes.) Using draft as the option will suppress the inclusion of the image in the output file and will replace the contents with the name of the image file that would have been seen. Using final will result in the image being placed in the output file. The default is final

3.16.2 Supported image format:

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.
We will discuss about this topic later.But right now what you have to know Using pdflatex will be usually much more simple for graphics inclusion as it supports widespread formats such as PDF, PNG and JPG. As knowing it is more than enough for now.

3.16.3 Including graphics:

Now that you have seen which formats you can include and how to 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}
```

You have to know that the arguments in the square brackets are optional,whereas arguments in the curly brackets are compulsory **Variety of possible attributes we can do with a graphics/figure:**
The variety of possible attributes that can be set is fairly large, so only the most common are covered below:

width=xx	Specify the preferred width of the imported image to xx.Whre xx=magnitude of length with proper unit.
height=xx	Specify the preferred height of the imported image to xx.
keepaspectratio	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, so that neither width nor height are exceeded.
scale=xx	Scales the image by the desired scale factor. e.g, 0.5 to reduce by half, or 2 to double.
angle=xx	This option can rotate the image by xx degrees (counter-clockwise)
trim=l b r t	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.
clip	For the trim option to work, you must set clip=true.
page=x	If the image file is a pdf file with multiple pages, this parameter allows you to use a different page than the first.

Note: Only specifying either width or height will scale the image whilst maintaining the aspect ratio.

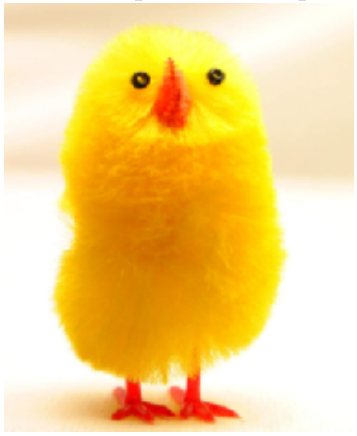
In order to use more than one option at a time, you have to separate each of them with a comma. The order the user gives the options matters. E.g If you want to

rotate a picture as well as specifying its width we should first rotate our 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".We will later discuss that This process(i.e inserting the picture) can be disrupted by layout. So, you need to learn some techniques to avoid this.

Examples:
A simple example:

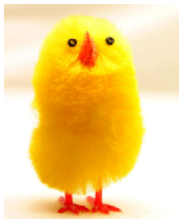
```
\includegraphics{Chick}
```

This will produce output like:



This simply imports the image, without any other processing. But if you want to import a larger picture you have to scale it down.

```
\includegraphics[scale=0.5]{Chick}
```



However, we will scale down this picture to illustrate the process of scaling down.

One can also specify the scale with respect to the width of a line in the local

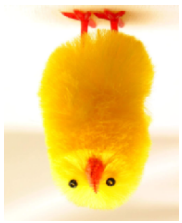
environment **(\linewidth)**, the width of the text on a page **(\textwidth)** or the height of the text on a page **(\textheight)** (pictures are not shown).

Note: In both the examples,we did not mention the file extension name of the picture, i.e the picture format:-whether it is .jpg,.png or .pdf. However, mentioning the file extension name is not necessary.

Now, we will illustrate an example to show how to use multiple arguments.

```
\includegraphics[scale=0.5, angle=180]{Chick}
```

The output would be:



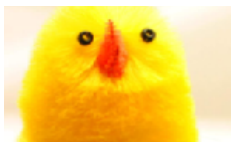
Here, we rotate the picture to 180°as well as scale it down to half.
And finally, an example of how to crop an image:
You have to use the command:

```
\includegraphics[trim=left bottom right top]{Image name}
where we have to provide the lengths for trim option pa-
rameters i.e for left,bottom,right & top.
```

Suppose,we use the command:

```
\includegraphics[trim = 5mm 30mm 5mm 5mm, clip, width=3cm]{Chick}
```

The output would be like:



Note: Here we declare/use multiple arguments i.e we trim the picture as well as set the clip option true for the trim option to work. We mention the desired width for the picture too.

Spaces in names: If the image file were called "Chick picture.png" instead of "Chick",you need to include the full filename when importing the image,i.e you have to use:

```
\includegraphics{Chick picture.png}
```

instead of

```
\includegraphics{Chick}
```

One option is to not use spaces in file names while you are importing a picture is to replace space with underscore,i.e to use the command:

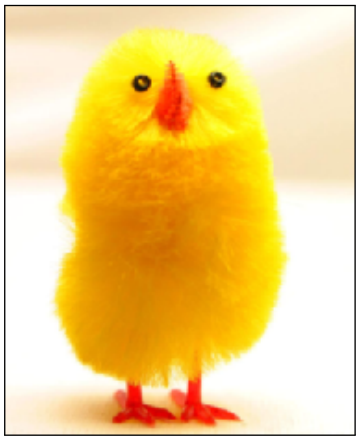
```
\includegraphics{Chick_picture.png}
```

Borders: It is possible to have LaTeX create a border around your image by using **\fbox**. you can also control the border padding with the **\setlength \fboxsep{0pt}** command. opt for the case to avoid any padding, so that the border will be placed tightly around the image. However you can also set it to 0.5pt,1pt or 2pt or to some other options according to our choice. The thickness of the border is adjusted by **\setlength\fboxrule{0.5pt}** command. however youcan also change 0.5pt accoring to your choice.

Now we will illustrate two examples for this portion. First, with a border which will be placed tightly around the image and second, with a border which will not be placed tightly around the image. So that, we could understand this topic properly.

```
\setlength\fboxsep{0pt}
\setlength\fboxrule{0.5pt}
\fbox{\includegraphics{Chick}}
```

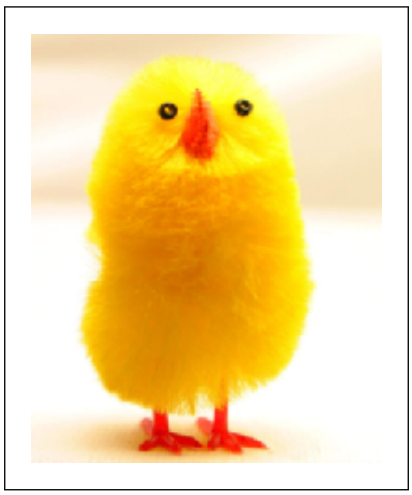
The output with respect to these commands would be like:



Now, clearly this picture has a tight border. Now, if we write:

```
\setlength\fboxsep{10pt}
\setlength\fboxrule{0.5pt}
\fbox{\includegraphics{Chick}}
```

The output would be like:



We have used 10pt here. So that, we could understand the difference properly.

3.17 Treating Figures as floats & providing Captions:

Now, we have discussed how to import graphics in LaTeX.However,just having a picture stuck in between paragraphs does not look professional. So, you need to learn a way of adding captions to a figure. Also, we have previously discussed that the importing graphics process can be disrupted by the page layout. So, if you just use the command:

```
\includegraphics[Optional arguments]{Picture name}
```

It will not gives you the desired output all the time.

An example: Suppose, you want a picture between two paragraphs, but you get it at the top of the next page.So, you need to learn a way so you could manage pictures into its desired position efficiently.This is where floats come into play.

3.17.1 Float:

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes "table" and "figure" floats, but we can also define new ones of our own.
Floats are there to deal with the problem of the object that won't fit on the present page, and to help when we 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 we specifies). They always have a caption describing them and they are always numbered so they can be referred to from elsewhere in the text. LaTeX automatically floats Tables and Figures, depending on how much space is left on the page at the point that they are processed. 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.(that is exactly what we are talking about in the previous example).
Suppose, you have many floats which will occur in rapid succession. Now, the main problem is that how they are supposed to fit on the page as well as leaving room for the text. In this case, LaTeX stacks them all up and prints them together if possible, or leaves them to the end of the chapter in protest.So, you need to learn some techniques to space them out within your text so that they intrude

neither on the thread of your argument or discussion, nor on the visual balance of the typeset pages.

3.17.2 Figure:

To create a figure that floats, you need to use the figure environment:

```
\begin{figure}[placement specifier]
... figure contents ...
\end{figure}
```

Now, we need to know about placement specifiers.
Definition of Placement specifier
It is a parameter exists as a compromise, and its purpose to give us the greater degree of control over where certain floats are placed.

Specifier	Permission
h	Place the float here, i.e., approximately at the same point it occurs in the source text (however, not exactly at the spot)
t	Position at the top of the page.
b	Position at the bottom of the page.
p	Put on a special page for floats only.
!	Override internal parameters LaTeX uses for determining "good" float positions.
H	Places the float at precisely the location in the LaTeX code. Requires the float package. This is somewhat equivalent to h!.

For doing some operations to a float, (Like using H as a placement specifier for a figure) we need to include the float package, i.e we have to mention the following command in the preamble of our document:

```
\usepackage{float}
```

Creating a list of figures at the beginning of document
To add a list of figures at the beginning of the document, we need to use the following command:

```
\listoffigures
```

Figure with borders: It's possible to get a thin border around all figures. You have to write the following once at the beginning of the document:

```
\usepackage{float}
\floatstyle{boxed}
\restylefloat{figure}
```

The border will not include the caption.

3.17.3 Caption:

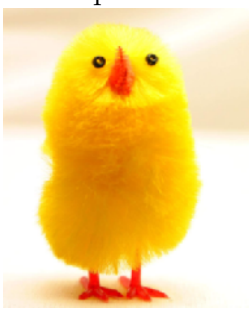
It is always good practice to add a caption to any figure or table. Fortunately, this is very simple in LaTeX . All you need to do is use the following command:

```
\caption{''text''}
```

in the float environment. Because of how LaTeX deals sensibly with logical structure, it will automatically keep track of the numbering of figures, so you do not need to include this within the caption text.
The location of the caption is traditionally underneath the float. However, it is up to you to therefore insert the caption command after the actual contents of the float (but still within the environment). If you place it before, then the caption will appear above the float.
Now, we will illustrate the process of giving a caption by providing an example:

```
\begin{figure}[h!]
\caption{A picture of a chick}
\centering
\includegraphics[scale=0.70]{Chick}
\end{figure}
```

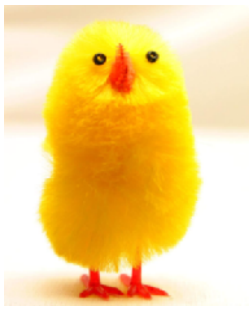
Figure 1: A picture of a chick



This would be the output of the given code.

Note: Now the placement specifier h has changed to ht. Here we use the **\caption{}** before the **\includegraphics** command. So, We get the caption before the picture. If we use **\caption{}** after **\includegraphics** then we get the caption after the picture. Now, we use **\centering** so that we get the picture at the middle of the page.In the example below, We will illustrate what will happen if we dont use **\centering**.

Figure 2: A picture of a chick



This will happen if we dont use **\centering**.
However, the \centering command can be replaced by the **center** environment. Then the code structure would be like:

```
\begin{figure}[placement specifier]
\begin{center}
```

continued to next page

continued from previous page

```
\includegraphics[Optional Arguments]{Picture name}
\caption{}
\end{center}
\end{figure}
```

Next example:
In this example, we will add the caption after the picture.

```
\begin{figure}[ht]
\centering
\includegraphics[scale=0.70]{Chick}
\caption{A picture of a chick}
\end{figure}
```

This will produce output like:

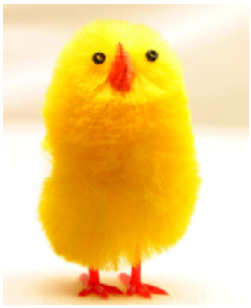


Figure 3: A picture of a chick

3.18 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.

There are several packages in LaTeX which can provide you the facilities of creating

beautiful tables. But as a beginner you need not to use those packages. By default, LaTeX provides you the facilities of creating tables using **tabular** environment.

3.18.1 The tabular environment:

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}
```

The table spec 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.

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
|	Vertical line
||	Double vertical line

If you want to create such a table, you cannot type '|' & '||' directly. You have to use $|$ & $\|$ commands for typing '|' & '||' respectively.
$:-This is generally called as math symbol. i.e for inserting any kind of math symbols you have to enclose that symbol with a opening $ and a closing $.

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 **\textwidth**.

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:

Letter	Meaning
b	bottom
c	center(default)
t	top

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.We will provide an example of it later.
In the first line you have pointed out how many columns you want, their alignment and the vertical lines to separate them. Once in the environment, you have to introduce the text you want, separating between cells and introducing new lines. The commands you have to use are the following:

&	Column Seperator
\\	start new row (additional space may be specified after\\ using square brackets, such as [6pt]
\hline	horizontal line
\newline	start a new line within a cell (in a paragraph column)
\cline{i-j}	partial horizontal line beginning in column i and ending in column j

Note: Any white space inserted between these commands is purely down to ones' preferences

Basic examples: This example shows how to create a simple table in LaTeX. It is a three-by-three table, but without any horizontal or vertical lines.

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

If you use the given code, the output would be like:

1	2	3
4	5	6
7	8	9

Note: In the code, we did not use \begin{center} & \end{center}. Otherwise the table would be created after the written line"If you use the given code, the output would be like" and the table creation would be started from the middle of the next line. Also, if we give \\ after the written line, the table creation would be started from the left side of the next line.

Expanding upon that by including some vertical lines:

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

If you use this code, the output would be:

1	2	3
4	5	6
7	8	9

Now, if you use two vertical lines as seperation lines between the columns,you have to use the following command:

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

This will produce output like:

1	2	3
4	5	6
7	8	9

A more complicated example will be provided where we will use both horizontal lines and vertical lines to seperate rows and columns respectively.

```
\begin{tabular}{|l|c|r|}
\hline
```

continued to next page

continued from previous page

```
1&2&3\\\hline
4&5&6\\\hline
7&8&9\\\hline
\end{tabular}
```

This will produce output like:

1	2	3
4	5	6
7	8	9

Now, we will illustrate an example where the leftmost column will be written in bold font and rightmost column will be written in the italics font. Remember the table where we specify the letters for the optional parameter **pos** and its role in creation of a tabl?The table has two columns -Letter and Meaning. We will rewrite the table where the 'Letter' column will be written in bold font and 'Meaning' column will be written in italics font.

Letter	*Meaning*
b	*bottom*
c	*center(default)*
t	*top*

The table is created by the following command:

```
\begin{center}
\begin{tabular}{|>{\bfseries}l|>{\itshape}r|}
\hline
Letter&Meaning\\\hline
b&bottom\\\hline
c&center(default)\\\hline
t&top\\\hline
\end{tabular}
\end{center}
```

Now,we will not discuss here about the arraypackage so you could learn the use of m{'width'} and b{'width}. But we will illustrate an example so you could understand the use of p{'width'}. Remember the table where we mention table specifications and its use. Now, if you look at the given code, you will see that we use p{10cm}. In the next example we will just replace the p{10cm} with r and you will see the difference.

```
\begin{center}
\begin{tabular}{|l|p{10cm}|}
\hline
Table specifications&Meaning\\\hline
l&left-justified column
\\\hline
c&centered column\\\hline
r&right-justified column\\\hline
p\{'width'\}&paragraph column with text vertically aligned at
the top\\\hline
m\{'width'\}&paragraph column with text vertically aligned in
the middle (requires array package)\\\hline
b\{'width'\}&paragraph column with text vertically aligned at
the bottom (requires array package)\\\hline
$\mid$&Vertical line\\\hline
$\parallel$&Double vertical line\\\hline
\end{tabular}
\end{center}
```

This will produce output like:

Table specifications	Meaning
l	left-justified column
c	centered column
r	right-justified column
p{'width'}	paragraph column with text vertically aligned at the top
m{'width'}	paragraph column with text vertically aligned in the middle (requires array package)
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array package)
|	Vertical line
||	Double vertical line

And if we just replace the p{width} with r in the code, we will get an output like:

Table specifications	M
l	left-justified c
c	centered c
r	right-justified c
p{'width'}	paragraph column with text vertically aligned at t
m{'width'}	paragraph column with text vertically aligned in the middle (requires array pa
b{'width'}	paragraph column with text vertically aligned at the bottom (requires array pa
|	Vertic
||	Double vertic

This will happen if we dont use p{width}. In an easy language, if you dont use it, the column will be of an undesired length which contains large amount of text. So, you have to use it for all those columns which will contain large amount of text.

Note: This problem is caused since LaTeX have some problems with the Text wrapping in tables issue. It is that LaTeX will not wrap text in cells, even if it overruns the width of the page.

Now, we will illustrate another example to make you learn the use of **\cline{i-j}**, i.e the use of partial horizontal line beginning in column i and ending in column j. If we use the following commands:

```
\begin{tabular}{|r|l|}
  \hline
  7C0 & hexadecimal \\
  3700 & octal \\ \cline{2-2}
  11111000000 & binary \\
  \hline \hline
  1984 & decimal \\
  \hline
\end{tabular}
```

The output would be like:

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

Manually Broken Paragraphs in Table Cells: Sometimes it is necessary to not rely on the breaking algorithm when using the p specifier, but rather specify the line breaks by hand. In this case it is easiest to use a \parbox:

```
\begin{tabular}{|c|c|}\hline
  boring cell content & \parbox[t]{4cm}{rather long par
  new par}
  \\\hline
\end{tabular}
```

The output would be like:

boring cell content	rather long par new par

Using this \parbox, what we have done that we make sure when the length of the text "rather long par new par" will cross 4cm, the linebreak will occur automatically. However, like we discussed before that the line breaking could be done by using \\ or \newline.

Other environment inside table: If you use LaTeX environment inside table cells, like **verbatim** or **enumerate**, you might encounter errors.

```
\begin{tabular}{c c}
        \hline
        \begin{verbatim}
        code
        \end{verbatim}
        & description
        \\ \hline
\end{tabular}
```

If you write this type of codes,you will get error.
To solve this problem, change column specifier to "paragraph" (p, m or b).Like:

```
\begin{tabular}{m{5cm} c}
```

An example: Suppose you want to create a table which provides about how a text can be changed into bold,italics,underlined and emphasized. If you use the following command:

```
\begin{center}
\begin{tabular}{|p{3cm}|p{7cm}|}\hline
Command&Use\\\hline
\begin{verbatim}{\bfseries{}}\end{verbatim}&To make a text
bold\\\hline
\begin{verbatim}{\textit{}}\end{verbatim}&To make a text
in italics\\\hline
\begin{verbatim}{\underline{}}\end{verbatim}&To make a
text underlined\\\hline
\begin{verbatim}{\emph{}}\end{verbatim}&To make a text
emphasized\\\hline
\end{tabular}
\end{center}
```

The output would be like

Command	Use
`{\bfseries{}}`	To make a text bold
`{\textit{}}`	To make a text in italics
`{\underline{}}`	To make a text underlined
`{\emph{}}`	To make a text emphasized

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 eight centered columns flanked by a single column on each side:

```
\begin{tabular}{l*{8}{c}r}
\hline
\#&Team&GP&W&D&L&GF&GA&GD&PTS\\\hline
1&Barcelona&17&15&1&1&49&12&37&46\\
2&Atletico Madrid&17&15&1&1&46&11&35&46\\
3&Real Madrid&17&13&2&2&49&21&28&41\\
4&Athletic&17&10&3&4&26&21&5&33\\\hline
\end{tabular}
```

This will produce output like:

#	Team	GP	W	D	L	GF	GA	GD	PTS
1	Barcelona	17	15	1	1	49	12	37	46
2	Atletico Madrid	17	15	1	1	46	11	35	46
3	Real Madrid	17	13	2	2	49	21	28	41
4	Athletic	17	10	3	4	26	21	5	33

3.19 Using table as a float:

The tabular environment that was used to construct the tables is not a float by default. Therefore, for tables you wish to float, wrap the tabular environment within a table environment, like this:

```
\begin{table}
  \begin{tabular}{...}
  ... table data ...
  \end{tabular}
\end{table}
```

If we want the table in the center, we have to use \begin{center} & \end{center}

An example of treating figure as a float:

```
\begin{table}[h!]
  \begin{center}
    \begin{tabular}{| l c r |}
    \hline
    1 & 2 & 3 \\
    4 & 5 & 6 \\
    7 & 8 & 9 \\
    \hline
    \end{tabular}
  \end{center}
  \end{table}
```

Here, we just want to show you the code structure if you want to use tables as a float.

Using table as a float also provides you the option to give captions to a table by using the command **\caption{}** inside the **table** environment.

3.20 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. 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. To use either package, include:

```
\usepackage{amsmath}
```

Or,

```
\usepackage{mathtools}
```

in the preamble of the document. The mathtools package loads the amsmath package and hence there is no need to \usepackage{amsmath} in the preamble if mathtools is used.(AMS:-American Mathematical Society)

3.20.1 The mathematics environment:

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, for example, I can say that a + a = 2a within this sentence.
- **displayed-**displayed formulas are separate from the main text.

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:

Type	Inline (within text) formulas	Displayed Equations	Displayed and automatically numbered equations
Environment	math	displaymath	equation
LaTeX shorthand	\(...\)	\[...\]	
TeX shorthand	$...$	$$...$$	
Comment			equation*(starred version) suppresses numbering, but requires amsmath

Note: Notice that, we leave empty cells in the table. It is done by leaving white spaces between two &s i.e between two column seperators. Also notice that we type $ int content of cells. Since this $ sign has a special role in LaTeX math mode, it cannot be typed directly. You have to use \$ for getting it typed in your document.

Suggestion: Using the $$...$$ should be avoided, as it may cause problems, particularly with the AMS-LaTeX macros. Furthermore, should a problem occur, the error messages may not be helpful.
The equation* and displaymath environments are functionally equivalent.
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:

1. 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.

2. Empty lines are not allowed. Only one paragraph per formula.

3. 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.

Note: In order for some operators, such as**\lim** or **\sum** to be displayed correctly inside some math environments, i.e inside a pair of $s. It might be convenient to write the **\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 will print a smaller $\sum$, but if we use **\displaystyle** in the environment i.e if we use **$\displaystyle \sum$** a big $\displaystyle \sum$ will be printed.(Like the summation sign we see in mathematical equations) However, it only works with AMSMATH package. It is also possible to force this behaviour for all math environments my declaring \everymath{\displaystyle} at the very beginning (i.e. before \begin{document}), rather useful in longer documents.

3.20.2 Mathematics symbols:

Mathematics has many symbols! One of the most difficult aspects of learning LaTeX is remembering how to produce symbols. There are of course a set of

symbols that can be accessed directly from the keyboard:
+ - = * / () ' :

However for using × and ÷ instead of '*' and '/' you have to use **$\times$** and **$\div$** respectively. For using >and <in text mode you have to use **\textgreater** and **\textless** respectively. However you could write them using $>$ and $<$ respectively. For using curly brackets and square brackets you have to check the 'Reserved characters' section.

- **Important Operators:**

Symbol	Command	Symbol	Command
$\pm$	\pm	$\mp$	\mp
$\times$	\times	$\div$	\div
$\frac{x}{y}$	{\frac{x}{y}}	$\sqrt{x}$	\sqrt{x}
$\sum$	\sum	$\int$	\int
$\cdot$	\cdot	$*$	\ast
$\cap$	\cap	$\cup$	\cup
$\sqcap$	\sqcap	$\sqcup$	\sqcap
$\forall$	\forall	x°	x^{\circ}
$\vee$	\vee	$\wedge$	\wedge
$\oplus$	\oplus	$\ominus$	\ominus
$\otimes$	\otimes	$\odot$	\odot
$\triangle$	\bigtriangleup	$\bigtriangledown$	\bigtriangledown

Now, we will discuss abot trigonometric operators.

- **Trigonometric Operators:**

sin	\sin	cos	\cos
tan	\tan	cot	\cot
sec	\sec	csc	\csc

Notice that for using csc, you have to use \csc instead of \cosec.However you can also use **\arcsin,\arccos & \arctan** instead of **\csc,\sec & \cot** respectively.

- **Hyperbolic Relations:**

Symbol	Command	Symbol	Command
sinh	\text{sinh}	cosh	\text{cosh}
tanh	\{tanh}	coth	\text{coth}
sech	\{sech}	cosech	\{cosech}

- **Important Relations:**

Symbol	Command	Symbol	Command
$\leq$	\le	$\geq$	\ge
$\neq$	\neq	$\sim$	\sim
$\ll$	\ll	$\gg$	\gg
$\doteq$	\doteq	$\simeq$	\simeq
$\subset$	\subset	$\supset$	\supset
$\subseteq$	\subseteq	$\supseteq$	\supseteq
$\approx$	\approx	*equiv*	\equiv
$\propto$	\propto	$\perp$	\perp
$\in$	\in	$\neq$	\ne
$\mid$	\mid	$\parallel$	\parallel

Negations of many of these relations can be formed by just putting \not before the symbol, or by slipping an n between the \ and the word. Here are a few examples, plus a few other negations; it works for many of the others as well.

Symbol	Command	Symbol	Command
$\nleq$	\nleq	$\ngeq$	\ngeq
$\nsim$	\nsim	$\ncong$	\ncong
$\nmid$	\nsim	$\nparallel$	\nparallel
$\not<$	\not<	$\not>$	\not>
$\not=$	\not=	$\not\sim$	\not\sim
$\not\le$	\not\le	$\not\ge$	\not\ge
$\not\approx$	\not\approx	$\not\cong$	\not\cong
$\not\equiv$	\not\equiv	$\not\parallel$	\not\parallel
$\not<$	\nless	$\not>$	\ngtr
$\lneq$	\lneq	$\gneq$	\gneq
$\lneqq$	\lneqq	$\gneqq$	\gneqq

Note: For using last mentioned symbols, loading **amsmath** package is not enough. You have to load **amssymb** package additionally in your preamble.

- **Greek Letters:**
  - **Lowercase Letters:**

Symbol	Command	Symbol	Command
α	\alpha	β	\beta
γ	\gamma	δ	\delta
ϵ	\epsilon	ε	\varepsilon
ζ	\zeta	η	\eta
θ	\theta	ϑ	\vartheta
ι	\iota	κ	\kappa
λ	\lambda	μ	\mu
ν	\nu	ξ	\xi
π	\pi	ω	\omega
ρ	\rho	ϱ	\varrho
σ	\sigma	ς	\varsigma
τ	\tau	υ	\upsilon
ϕ	\phi	φ	\varphi
χ	\chi	ψ	\psi

  - **Uppercase Letters:**

Symbol	Command	Symbol	Command
Γ	\Gamma	Δ	\Delta
Θ	\Theta	Λ	\Lambda
Ξ	\Xi	Π	\Pi
Σ	\Sigma	Φ	\Phi
Ψ	\Psi	Ω	\Omega

- **Arrows:**

Symbol	Command	Symbol	Command
$\leftarrow$	\gets	$\rightarrow$	\to
$\leftarrow$	\leftarrow	$\rightarrow$	\rightarrow
$\Leftarrow$	\Leftarrow	$\Rightarrow$	\Rightarrow
$\Leftrightarrow$	\Leftrightarrow	$\mapsto$	\mapsto
Symbol	Command	Symbol	Command
$\leftharpoonup$	\leftharpoonup	$\leftharpoondown$	\leftharpoondown
$\rightharpoonup$	\rightharpoonup	$\rightharpoondown$	\rightharpoondown
$\rightleftharpoons$	\rightleftharpoons	$\hookleftarrow$	\hookleftarrow
$\longleftarrow$	\longleftarrow	$\longrightarrow$	\longrightarrow
$\longleftarrow$	\Longleftarrow	$\longrightarrow$	\longRightarrow
$\longleftrightarrow$	\longleftrightarrow	$\longmapsto$	\longmapsto
$\hookrightarrow$	\hookrightarrow	$\leadsto$	\leadsto
$\uparrow$	\uparrow	$\downarrow$	\downarrow
$\Uparrow$	\Uparrow	$\Downarrow$	\Downarrow
$\updownarrow$	\updownarrow	$\Updownarrow$	\Updownarrow
$\nearrow$	\nearrow	$\searrow$	\searrow
$\nwarrow$	\nwarrow	$\swarrow$	\swarrow

- **Dots:**

Symbol	Command	Symbol	Command
$\ldots$	\ldots	$\vdots$	\vdots
$\cdots$	\cdots	$\ddots$	\ddots

- **Accents:**

Symbol	Command	Symbol	Command
$\hat{x}$	\hat{x}	$\grave{x}$	\grave{x}
$\dot{x}$	\dot{x}	$\ddot{x}$	\ddot{x}
$\acute{x}$	\acute{x}	$\grave{x}$	\grave{x}
$\tilde{x}$	\tilde{x}	$\mathring{x}$	\mathring{x}
$\bar{x}$	\bar{x}	$\vec{x}$	\vec{x}

Note: When applying accents to i and j, you can use \imath & \jmath to keep the dots from interfering with the accents: $\imath$ & $\jmath$

\tilde and **\hat** have wide versions that allow you to accent an expression:

symbol	Command	Symbol	Command
$\widehat{3+x}$	\widehat{3+x}	$\widetilde{abc}$	\widetilde{abc}

- **Others:**

Symbol	Command	Symbol	Command
∞	\infty	$\triangle$	\triangle
$\angle$	\angle	$\hbar$	\hbar
ℓ	\ell	$\mho$	\mho
$\prime$	\prime	$\emptyset$	\emptyset
∇	\nabla	$\surd$	\surd
∂	\partial	$\Re$	\Re
$\top$	\top	$\bot$	\bot
$\vdash$	\vdash	$\dashv$	\dashv
§	\S	$\Box$	\Box
©	\copyright	£	\pounds

Note: Dont forget to insert all the symbols in math mode i.e inside a pair of $s.

3.20.3 Examples:

- **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:

```
$\cos$ (2$\theta$) = $\cos$ $^2$ $\theta$ - $\sin$
$^2$ $\theta$
```

This will produce output like: $\cos\ (2\theta) = \cos\ ^2\ \theta$ - $\sin\ ^2\ \theta$
Is not it a little messy? The formula above can by typeset by the following command too:

```
$\cos (2\theta) = \cos^2 \theta - \sin^2 \theta$
```

And this is the right way to typeset such formulas in LaTeX.

Note: For writing a formula you dont have to insert every symbol in math mode,i.e you dont have to enclose every command within a pair of $s. Instead enclose the formula by a pair of $s

For certain operators such as limits, the subscript is placed underneath the operator:

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

This will produce output like: $\lim_{x \to \infty} \exp(-x) = 0$

For the modular operator there are two commands:**bmod** & **pmod**

```
$a \bmod b$
```

This will produce output like: $a \bmod b$

```
$x \equiv a \pmod b$
```

It's output: $x \equiv a \pmod b$

- **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 }).

```
$k_{n+1} = n^2 + k_n^2 - k_{n-1}$
```

  It's Output:$k_{n+1} = n^2 + k_n^2 - k_{n-1}$

  An underscore () can be used with a vertical bar (|) to denote evaluation using subscript notation in mathematics:

```
$f(n) = n^5 + 4n^2 + 2 |_{n=17}$
```

  It's ouput: $f(n) = n^5 + 4n^2 + 2|_{n=17}$

- **Fractions and Binomials:**
  A fraction is created using the **\frac{numerator}{denominator}** Command. (You remember the terms numerator and denominator, right? However, If not, numerator and denominator are top and bottom of a fractional number respectively). Likewise, the binomial coefficient may be written using the **\binom** command.

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

  It produces output like: $\frac{n!}{k!(n-k)!} = \binom{n}{k}$
  It is also possible to use the \ choose command instead of **\binom**. If you use the following command you will get the same output.

```
$\frac{n!}{k!(n-k)!} = {n \choose k}$
```

  You can embed fractions within fractions:

```
$\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$
```

It's output: $\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$
Note that when appearing inside another fraction, or in inline text $\frac{a}{b}$, a fraction is noticeably smaller than in displayed mathematics. The **\tfrac** and **\dfrac** commands force the use of the respective styles, **\textstyle & \displaystyle** . Similarly, the **\tbinom** and **\dbinom** commands typeset the binomial coefficient. t and d are referred to the **textstyle & displaystyle** respectively.
Another way to write the fraction is to use the **\over** command instead of **\frac** command.

```
${n! \over k!(n-k)!} = {n \choose k}$
```

This will produce output like: $\frac{n!}{k!(n-k)!} = \binom{n}{k}$

Noe both **\choose & \over** can work without asmmath package. So if you load asmmath package in your document by mentioning it in the preamble, you may get a warning message for using them.

For relatively simple fractions, especially within the text, it may be more aesthetically pleasing to use powers and indices:

```
$^3/_7$
```

This will provide output like: $^3/_7$
If this looks a little "loose" (overspaced), a tightened version can be defined by inserting some negative space:

```
\newcommand*\rfrac[2]{{}^{#1}\!/_{#2}}
$\rfrac{3}{7}$
```

This will produce output like: $^{3}/_{7}$

Note: If you need it only for once, then it's ok. But if you have to use them throughout the document, usage of xfrac package is recommended.This package provides **\sfrac** command to create slanted fractions.

Usages of sfrac:

```
Take $\sfrac{1}{2}$ cup of sugar, \dots
another example is:
$3\times\sfrac{1}{2}=1\sfrac{1}{2}$
```

Those commands will produce output like:
Take $^{1}/_{2}$ cup of sugar, ...
&
$3 \times {}^{1}/_{2} = 1{}^{1}/_{2}$

However, if you don't use **sfrac** command. You could use the following commands too:

```
Take ${}^1/_2$ cup of sugar, \dots
                    &
$3\times{}^1/_2=1{}^1/_2$
```

These commands will produce output like:
Take $^{1}/_{2}$ cup of sugar, ...
&
$3 \times {}^{1}/_{2} = 1{}^{1}/_{2}$

Note: Alternatively, the nicefrac package provides the **\nicefrac** command, whose usage is similar to **\sfrac**.

- **Continued fractions:**
  Continued fractions should be written using **\cfrac** command.

```
\begin{equation}
  x = a_0 + \cfrac{1}{a_1
          + \cfrac{1}{a_2
          + \cfrac{1}{a_3 + \cfrac{1}{a_4} } } }
\end{equation}
```

Output would be like:

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}} \tag{1}$$

Note: Notice that, LaTeX provides numbering for equations if you use equation environment. However, if you use **\begin{equation*} and \end{equation*}** you can omit the numbering. For multiline equations, LaTeX provides another environment named **align**, i.e you have to wrap the equations by **\begin{align} & \end{align}**. However, it also provides some numbering against equations. To omit this you have to use: **\begin{align*} & \end{align*}**

- **Multiplication of two numbers:**
  An example:

```
\begin{equation}
\frac{
    \begin{array}[b]{r}
```

continued to next page

continued from previous page

```
 \left( x_1 x_2 \right)\\
      \times \left( x'_1 x'_2 \right)
    \end{array}
  }{
    \left( y_1y_2y_3y_4 \right)
  }
\end{equation}
```

This will produce output like:

$$\frac{\begin{array}{c}\left(x_1 x_2 \right)\\ \times \left(x'_1 x'_2 \right)\end{array}}{\left(y_1y_2y_3y_4 \right)} \tag{2}$$

- **Roots:**
  The **\sqrt** command creates a square root surrounding an expression. It accepts an optional argument specified in square brackets (

  and

) to change magnitude:

```
$\sqrt{\frac{a}{b}}$
```

This will produce output like: $\sqrt{\frac{a}{b}}$
You have to learn how to write nth root. A Simpleexample of writing nth root:

```
$\sqrt[b]{a}$
```

It's output: $\sqrt[b]{a}$
A complicated example:

```
$\sqrt[n]{1+x+x^2+x^3+\ldots}$
```

The output would be like: $\sqrt[n]{1+x+x^2+x^3+\ldots}$

Some people prefer writing the square root "closing" it over its content. This method arguably makes it more clear what is in the scope of the root sign. This habit is not normally used while writing with the computer, but if you still want to change the output of the square root, LaTeX gives you this possibility. Just add the following code in the preamble of your document:

```
% New definition of square root:
% it renames \sqrt as \oldsqrt
\let\oldsqrt\sqrt
% it defines the new \sqrt in terms of the old one
\def\sqrt{\mathpalette\DHLhksqrt}
\def\DHLhksqrt#1#2{%
\setbox0=\hbox{$#1\oldsqrt{#2\,}$}\dimen0=\ht0
\advance\dimen0-0.2\ht0
\setbox2=\hbox{\vrule height\ht0 depth -\dimen0}%
{\box0\lower0.4pt\box2}}
```

Like it is mentioned, by adding the following code in the preamble of your document, you are defining the new **\sqrt** in terms of the old one, i.e in terms of **\oldsqrt**. The new square root can be seen in the picture on the

$$\sqrt{\frac{a}{b}} \qquad \sqrt{\frac{a}{b}}$$

left, compared to the old one on the right in the picture above.

Note: Unfortunately this code which is shown above, won't work if you want to use multiple roots: if you try to write $\sqrt[b]{a}$ as **\sqrt[b]{a}** after you used the code above, you'll just get a wrong output. In other words, you can redefine the square root this way only if you are not going to use multiple roots in the whole document.

An alternative piece of TeX code that does allow multiple roots is:

```
\LetLtxMacro{\oldsqrt}{\sqrt} % makes all sqrts closed
\renewcommand{\sqrt}[1][\ ]{%
\def\DHLindex{#1}\mathpalette\DHLhksqrt}
\def\DHLhksqrt#1#2{%
\setbox0=\hbox{$#1\oldsqrt[\DHLindex]{#2\,}$}\dimen0=
\ht0
\advance\dimen0-0.2\ht0
\setbox2=\hbox{\vrule height\ht0 depth -\dimen0}%
{\box0\lower0.71pt\box2}}
```

After doing this, if you use the following commands
$\sqrt[a]{b} \quad \oldsqrt[a]{b}$
It will produce output like:

$$\sqrt[a]{b} \quad \sqrt[a]{b}$$

However this requires to load the **letltxmacro** package in the preamble of your document.

- **Sums and integrals:**
  The **\sum** and **\int** commands insert the sum and integral symbols respectively, with limits specified using the caret () and underscore (). The typical notation for sums is:

```
$\sum_{i=1}^{10} t_i$
```

This will produce output like: $\sum_{i=1}^{10} t_i$
The limits for the integrals follow the same notation. It's also The important to represent the integration variables with an upright d, which in math mode is obtained through the **\mathrm{}** command, and with a small space separating it from the integrand, which is attained with the \, command.

```
$\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$
```

It will produce output like: $\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$
There are many other "big" commands which operate in a similar manner:

Symbol	command	Symbol	Command
$\sum$	\sum	$\prod$	\prod
$\coprod$	\coprod	$\bigoplus$	\bigoplus
$\bigotimes$	\bigotimes	$\bigodot$	\bigodot
$\bigcap$	\bigcap	$\bigcup$	\bigcup
$\bigsqcup$	\bigsqcup	$\biguplus$	\biguplus
$\bigvee$	\bigvee	$\bigwedge$	\bigwedge
$\int$	\int	$\oint$	\oint
$\iint$	\iint	$\iiint$	\iiint
$\iiiint$	\iiiint	$\int \cdots \int$	\idotsint

For more integral symbols, including those not included by default in the Computer Modern font, try the **esint** package.

The **\substack** command allows the use of \\ to write the limits over multiple lines:

```
\begin{equation*}
\sum_{\substack{
   0<i<m \\
   0<j<n
  }}
 P(i,j)
\end{equation*}
```

The output would be like:

$$\sum_{\substack{0<i<m \\ 0<j<n}} P(i,j)$$

If you want the limits of an integral to be specified above and below the symbol (like the sum), use the \limits command:

```
$int\limits_a^b$
```

This will produce output like: $\int\limits_a^b$

However if you want this to apply to ALL integrals, it is preferable to specify the **intlimits** option when loading the **amsmath** package:
\usepackage[intlimits]{amsmath}
Subscripts and superscripts in other contexts as well as other parameters to amsmath package related to them are described later.
For bigger integrals, you may use personal declarations, or the **bigints** package.

- **Brakets,braces and delimeters** The use of delimiters such as brackets soon becomes important when dealing with anything but the most trivial equations. Without them, formulas can become ambiguous. Also, special types of mathematical structures, such as matrices, typically rely on delimiters to enclose them.
  There are a variety of delimiters available for use in LaTeX

```
$
( a ), [ b ], \{ c \}, | d |, \| e \|,
\langle f \rangle, \lfloor g \rfloor,
\lceil h \rceil, \ulcorner i \urcorner$
```

Output:
$(a), [b], \{c\}, |d|, \|e\|, \langle f \rangle, \lfloor g \rfloor, \lceil h \rceil, \ulcorner i \urcorner$

- **Automatic Sizing:**
  Very often mathematical features will differ in size, in which case the delimiters surrounding the expression should vary accordingly. This can be done automatically using the **\left, \right, and \middle** commands. Any of the previous delimiters may be used in combination with these:

```
$\left(\frac{x^2}{y^3}\right)$
```

This will produce output like: $\left(\frac{x^2}{y^3}\right)$

```
$P\left(A=2\middle|\frac{A^2}{B}>4\right)$
```

It's output :$P\left(A=2\middle|\frac{A^2}{B}>4\right)$
Curly braces are defined differently by using **\left\{ and \right\}**.

```
$\left\{\frac{x^2}{y^3}\right\}$
```

It's output:$\left\{\frac{x^2}{y^3}\right\}$
If a delimiter on only one side of an expression is required, then an invisible delimiter on the other side may be denoted using a period (.).

```
$\left.\frac{x^3}{3}\right|_0^1$
```

It's output: $\left.\frac{x^3}{3}\right|_0^1$

- **Manual sizing:**
  In certain cases, the sizing produced by the **\left** and **\right** commands may not be desirable, or you may simply want finer control over the delimiter sizes. In this case, the **\big**, **\Big, \bigg** and **\Bigg** modifier commands may be used:

Symbol	Command	Symbol	Command
(	\big(	(	\Big(
(	\bigg(	(	\Bigg(

A comparison of typesets of those commands and normal sized opening brace:

```
$( \big( \Big( \bigg( \Bigg($
```

Output: $(\big(\Big(\bigg(\Bigg($

– **Usefulness of \big,\Big,\bigg & \Bigg:**

These commands are primarily useful when dealing with nested delimiters. For example, when typesetting

```
$\frac{\mathrm d}{\mathrm d x} \left( k g(x) \right)$
```

It's output: $\frac{\mathrm d}{\mathrm d x} \left(k g(x) \right)$
we notice that the **\left** and **\right** commands produce the same size delimiters as those nested within it. This can be difficult to read. To fix this, we write

```
$\frac{\mathrm d}{\mathrm d x} \big( k g(x) \big)$
```

It will produce output like : $\frac{\mathrm{d}}{\mathrm{d}x}\Big(kg(x)\Big)$

Notice the difference carefully. In the typeset of the second code, the size of opening brackets are different. One thing should be mentioned for showing you the difference properly, we have used **\Large{}** to enlarge the both outputs.

- **Matrices and arrays:**
  A basic matrix may be created using the matrix environment[3]: in common with other table-like structures, entries are specified by row, with columns separated using an ampersand (&) and a new rows separated with a double backslash (\\)

```
$\begin{matrix}
  a & b & c \\
  d & e & f \\
  g & h & i
 \end{matrix}$
```

will produce output like: $\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$

To specify alignment of columns in the table, use starred version:

```
$
\begin{matrix}
  -1 & 3 \\
  2 & -4
 \end{matrix}
 =
 \begin{matrix}
  -1 & 3 \\
  2 & -4
 \end{matrix}$
```

Output:

$$\begin{matrix} -1 & 3 \\ 2 & -4 \end{matrix} = \begin{matrix} -1 & 3 \\ 2 & -4 \end{matrix}$$

The alignment by default is **c**(center) but it can be any column type valid in array environment.

The alignment by default is c but it can be any column type valid in array environment.

Environment	Surrounding delimeter	Note
pmatrix	()	centers columns by default
pmatrix*	()	allows to specify alignment of columns in optional parameter
bmatrix	[]	centers columns by default
bmatrix*	[]	allows to specify alignment of columns in optional parameter
Bmatrix	{}	centers columns by default
Bmatrix*	{}	allows to specify alignment of columns in optional parameter
vmatrix	||	centers columns by default
vmatrix*	||	allows to specify alignment of columns in optional parameter
Vmatrix	|||	centers columns by default.
Vmatrix*	|||	allows to specify alignment of columns in optional parameter

When writing down arbitrary sized matrices, it is common to use horizontal, vertical and diagonal triplets of dots (known as ellipses) to fill in certain columns and rows. These can be specified using the **\cdots, \vdots and \ddots** respectively..

```
$
A_{m,n} =
 \begin{pmatrix}
  a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
```

continued to next page

continued from previous page

```
  a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  a_{m,1} & a_{m,2} & \cdots & a_{m,n}
 \end{pmatrix}$
```

It's output:- $A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$

However,knowing the use of pmatrix*, bmatrix*, Bmatrix*, vmatrix*, Vmatrix* is not necessary for now.

```
$
\begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
$
```

It will produce output like: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

```
$
\begin{Bmatrix}
a & b \\
c & d
\end{Bmatrix}
$
```

It will produce output like: $\begin{Bmatrix} a & b \\ c & d \end{Bmatrix}$

```
$
\begin{vmatrix}
a & b \\
c & d
\end{vmatrix}
$
```

This will produce output like: $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$

```
$
\begin{Vmatrix}
a & b \\
c & d
\end{Vmatrix}
$
```

This will produce output like: $\begin{Vmatrix} a & b \\ c & d \end{Vmatrix}$

You might be wondering that why we provide a seperate example for the pmatrix environment. That is to make you understand how to write a big matrix using **\vdots,\ddots,\cdots**.

In some cases you may want to have finer control of the alignment within each column, or want to insert lines between columns or rows. This can be achieved using the array environment, which is essentially a math-mode version of the tabular environment, which requires that the columns be pre-specified:

```
$
\begin{array}{c|c}
  1 & 2 \\
```

continued to next page

continued from previous page

```
  \hline
 3 & 4
\end{array}
$
```

This will produce output like: $\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}$

Using fraction as an element of matrix:

```
$
M = \begin{bmatrix}
       \frac{5}{6} & \frac{1}{6} & 0           \\
       \frac{5}{6} & 0           & \frac{1}{6} \\
       0           & \frac{5}{6} & \frac{1}{6}
     \end{bmatrix}

$
```

It will produceoutput like: $M = \begin{bmatrix} \frac{5}{6} & \frac{1}{6} & 0 \\ \frac{5}{6} & 0 & \frac{1}{6} \\ 0 & \frac{5}{6} & \frac{1}{6} \end{bmatrix}$

Notice that, AMS matrix class of environments doesn't leave enough space when used together with fractions resulting in output similar to this. To counteract this problem, add additional leading space with the optional parameter to the \\ command.

```
$
M = \begin{bmatrix}
     \frac{5}{6} & \frac{1}{6} & 0           \\[0.3em]
```

continued to next page

continued from previous page

```
        \frac{5}{6} & 0& \frac{1}{6} \\[0.3em]
        0           & \frac{5}{6} & \frac{1}{6}
      \end{bmatrix}
$
```

It will produce output like: $M = \begin{bmatrix} \frac{5}{6} & \frac{1}{6} & 0 \\ \frac{5}{6} & 0 & \frac{1}{6} \\ 0 & \frac{5}{6} & \frac{1}{6} \end{bmatrix}$

Now, notice the spacing between two elements. That's the difference.

If you need "border" or "indexes" on your matrix, plain TeX provides the macro **\bordermatrix**

```
$
M = \bordermatrix{~ & x & y \cr
                  A & 1 & 0 \cr
                  B & 0 & 1 \cr}
$
```

It will produce output like: $M = \begin{array}{c} \\ A \\ B \end{array} \begin{array}{c} \begin{array}{cc} x & y \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{array}$

- **Matrices in running text:**
  To insert a small matrix, and not increase leading in the line containing it, use **smallmatrix** environment:
  A matrix in text must be set smaller:

  clearpage

```
$\bigl(\begin{smallmatrix}
a&b\\ c&d
\end{smallmatrix} \bigr)$
to not increase leading in a portion of text.
```

$\bigl(\begin{smallmatrix} a&b\\ c&d \end{smallmatrix} \bigr)$ to not increase leading in a portion of text.

- **Adding text to the equation:**
  The math environment differs from the text environment in the representation of text. Here is an example of trying to represent text within the math environment:

```
$
50 apples \times 100 apples = lots of apples^2
$
```

It will produce output like:

$$50apples \times 100apples = lotsofapples^2$$

There are two noticeable problems: there are no spaces between words or numbers, and the letters are italicized and more spaced out than normal. Both issues are simply artifacts of the maths mode, in that it treats it as a mathematical expression: spaces are ignored (LaTeX spaces mathematics according to its own rules), and each character is a separate element (so are not positioned as closely as normal text).

There are a number of ways that text can be added properly. The typical way is to wrap the text with the **\text{...}** command(a similar command is **\mbox{...}**, though this causes problems with subscripts, and has a less descriptive name). Let's see what happens when the above equation code is adapted:

```
$
50 \text{apples} \times 100 \text{apples}
 = \text{lots of apples}^2
```

```
$
```

It will produce output like:

$$50\text{apples} \times 100\text{apples} = \text{lots of apples}^2$$

Now,The text looks better. However, there are no gaps between the numbers and the words. Unfortunately, you are required to explicitly add these. There are many ways to add spaces between maths elements, but for the sake of simplicity we may simply insert space characters into the **\text** commands.
i.e to use the command:

```
$
50 \text{ apples} \times 100 \text{ apples}
 = \text{lots of apples}^2
$
```

It will produce output like:

$$50 \text{ apples} \times 100 \text{ apples} = \text{lots of apples}^2$$

- **Formatted text:**
  Using the **\text** is fine and gets the basic result. Yet, there is an alternative that offers a little more flexibility. You may recall the introduction of font formatting commands, such as **\textrm, \textit, \ textbf**, etc. These commands format the argument accordingly, e.g., **\textbf{bold text}** gives bold text. These commands are equally valid within a maths environment to include text. The added benefit here is that you can have better control over the font formatting, rather than the standard text achieved with **\text**.

```
$
50 \textrm{ apples} \times 100
 \textbf{ apples} = \textit{lots of apples}^2
$
```

It will produce output like:

$$50 \text{ apples} \times 100 \textbf{ apples} = \textit{lots of apples}^2$$

However, in the case of LaTeX there are more than one way to skin a cat! There are a set of formatting commands very similar to the font formatting ones just used, except they are specifically for text in math mode.

- **Formatting Math Symbol:** Like, we previously mentioned they are specifically at text in math mode. (You need to load **amsfonts** package in your preamble.)

LaTeX command	Sample	Description	Common Use
\mathnormal{...}	$ABCDEFabcdef123456$	the default math font	most mathematical notation
\mathrm{...}	ABCDEFabcde123456	this is the default or normal font, unitalicised	units of measurement, one word functions
\mathit{...}	$\mathit{ABCDEFabcde123456}$	italicised font	
\mathbf{...}	**ABCDEFabcde123456**	bold font	vectors
\mathsf{...}	ABCDEFabcde123456	sans-serif	
\mathtt{...}	`ABCDEFabcde123456`	Monospace (fixed width font)	

continued to next page

continued from previous page

\mathcal{...}	$\mathcal{ABCDEFGHIJKL}$	Calligraphy (Upper case only)	often used for sheaves/schemes and categories, used to denote cryptological concepts like an alphabet of definition($\mathcal{A}$), message space ($\mathcal{M}$), ciphertext space($\mathcal{C}$) etc.
\mathfrak{...}	$\mathfrak{ABCDEFabcde123456}$	FRAKTUR	Almost canonical font for Lie algebras, with superscript used to denote New Testament papyri, ideals in ring theory
\mathbb{...}	$\mathbb{ABCDEF}\nexists\nvDash\nVdash\not\supset\not\subseteq\not\subset$	Blackboard bold	Used to denote special sets (e.g. real numbers)
\mathscr{...}		Script	An alternative font for categories and sheaves

\mathbb{...} are genrally used to typeset set letters. Set letters are the letters which are commonly used in maths to name special sets like the sets of natural numbers. It works with the asmsymb package and works properly with capital letters only.(That's why we get undesired output for **\mathbb{abcde123456}** i.e we get output as ∋⊮⊭⊯⫋⫋⫋ for using this.
Set of natural numbers: $\mathbb{N}$
Set of real numbers : $\mathbb{R}$
Set of integers : $\mathbb{Z}$
Set of rational numbers: $\mathbb{Q}$
Set of complex numbers: $\mathbb{C}$

The maths formatting commands can be wrapped around the entire equation, and not just on the textual elements: they only format letters, numbers, and uppercase Greek, and the rest of the maths syntax is ignored.
To bold lowercase Greek or other symbols use the **\boldsymbol** command; this will only work if there exists a bold version of the symbol in the current font. As a last resort there is the **\pmb** command(poor mans bold): this prints multiple versions of the character slightly offset against each other

```
$
\boldsymbol{\beta} = (\beta_1,\beta_2,\dotsc,\beta_n)
$
```

It's output:
$\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_n)$
You can also change the size of the fonts in math mode. We will discuss it later.

- **Controlling horizontal spacing:**
LaTeX is obviously pretty good at typesetting maths—it was one of the chief aims of the core TeX system that LaTeX extends. However, it can't always be relied upon to accurately interpret formulas in the way you did. It has to make certain assumptions when there are ambiguous expressions. The result tends to be slightly incorrect horizontal spacing. In these events, the output is still satisfactory, yet any perfectionists will no doubt wish to fine-tune their formulas to ensure spacing is correct. These are generally very subtle

adjustments.
There are other occasions where LaTeX has done its job correctly, but you just want to add some space, maybe to add a comment of some kind. For example, in the following equation, it is preferable to ensure there is a decent amount of space between the maths and the text.

```
\begin{equation*}
\[ f(n) = \left\{
  \begin{array}{l l}
    n/2 & \quad \text{if $n$ is even}\\
    -(n+1)/2 & \quad \text{if $n$ is odd}
  \end{array} \right.\]
\end{equation*}
```

This will produce output like:

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ -(n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

This code produces errors with Miktex 2.9 and does not yield the results seen on the right. Use **\textrm** instead of just **\text**. However, this is an error which is compiler depended. So, using **\textrm** is not compulsory in this type of code.However, you could also write **\mbox** instead of **\text** in that code.
LaTeX has defined two commands that can be used anywhere in documents (not just maths) to insert some horizontal space. They are **\quad** and **\qquad**.
A **\quad** is a space equal to the current font size. So, if you are using an 11pt font, then the space provided by **\quad** will also be 11pt (horizontally, of course.)
The **\qquad** gives twice that amount. As you can see from the code from the above example, **\quad** were used to add some separation between the maths and the text.
A good example for understanding the use of **\quad** would be displaying the simple equation for the indefinite integral of y with respect to x:
$\int y \, \mathrm{d}x$
If you were to try this, you may write:

```
$\int y \mathrm{d}x$
```

It's output: $\int y\mathrm{d}x$

As you can see, LaTeX doesn't respect the white-space left in the code to signify that the y and the dx are independent entities.Instead, it lumps them altogether. A **\quad** would clearly be overkill in this situation—what is needed are some small spaces to be utilized in this type of instance, and that's what LaTeX provides:

Command	Description	Size
\,	Small space	3/18 of a quad.
\:	Medium Space	4/18 of a quad
\;	Large Space	5/18 of a quad.
\!	negative Spacce	-3/18 of a quad

So, to rectify the current problem you can use those commands. Now, we will give an example by using those commands to rectify the problem.

```
$\int y\, \mathrm{d}x$
```

It will produce output like: $\int y\, \mathrm{d}x$
If we use:

```
$\int y\: \mathrm{d}x$
```

It will produce output like: $\int y\: \mathrm{d}x$
And if we use:

```
$\int y\; \mathrm{d}x$
```

It will produce output like: $\int y \, \mathrm{d}x$

Use of Negative Space:
The negative space may seem like an odd thing to use, however, it wouldn't be there if it didn't have some use! Take the following example:

```
$
\left(
    \begin{array}{c}
      n \\
      r
    \end{array}
  \right) = \frac{n!}{r!(n-r)!}
$
```

It will produce output like: $\left(\begin{array}{c} n \\ r \end{array} \right) = \frac{n!}{r!(n-r)!}$

The matrix-like expression for representing binomial coefficients is too padded. There is too much space between the brackets and the actual contents within. This can easily be corrected by adding a few negative spaces after the left bracket and before the right bracket.

```
$
\left(\!
    \begin{array}{c}
     n \\
      r

    \end{array}
  \!\right) = \frac{n!}{r!(n-r)!}
 $
```

It will produce output like: $\left(\begin{array}{c} n \\ r \end{array} \right) = \frac{n!}{r!(n-r)!}$

Upto this we can call the basics of creating a article using LaTeX .

4 References:

- http://en.wikibooks.org/wiki/LaTeX
- tex.stackexchange.com