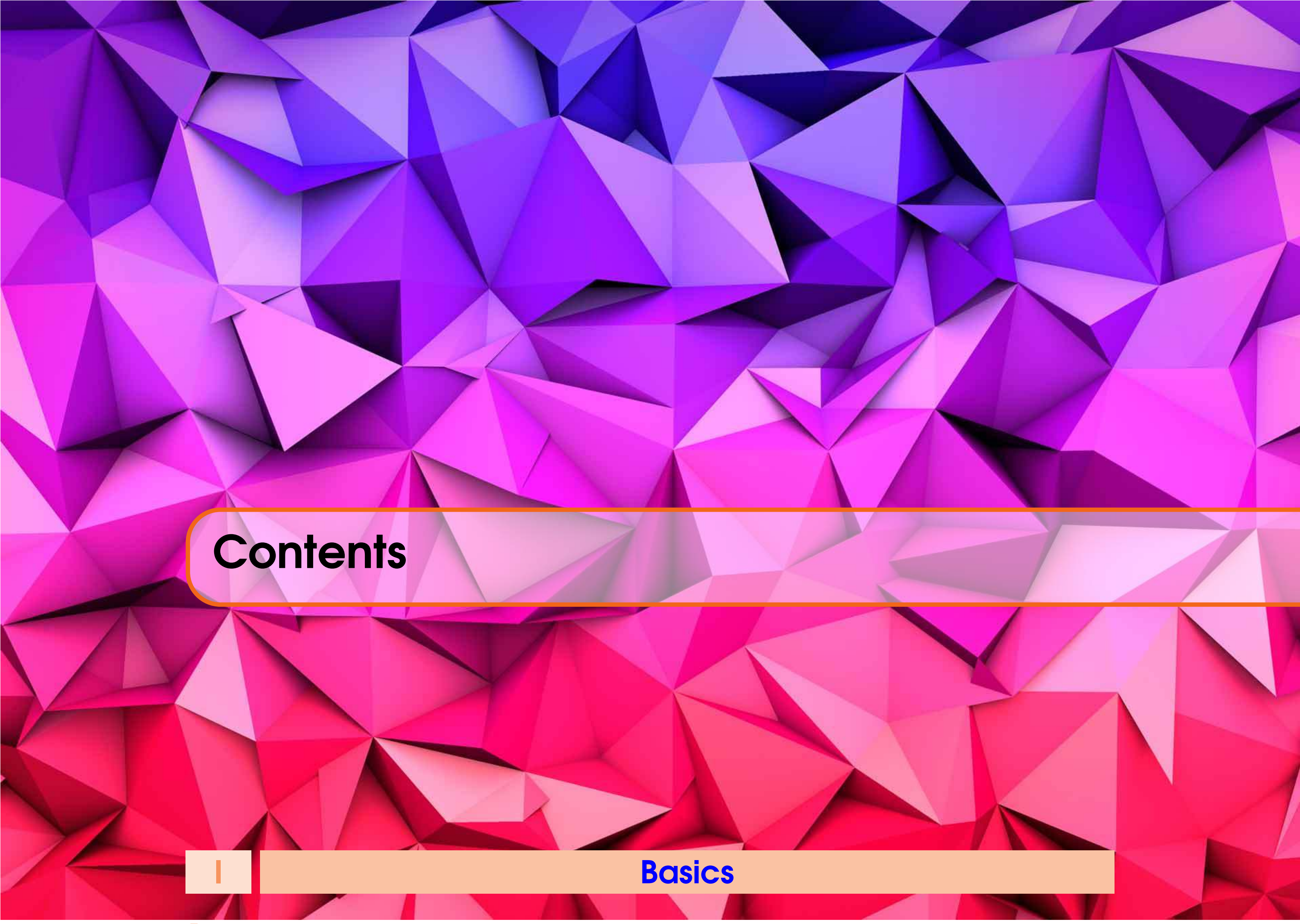
LaTex Document

A Pocket Reference



Contents

Basics

1. Introduction of *LaTeX*

1.1 TEX

TeX is a low-level markup and programming language created by Donald Knuth to typeset documents attractively and consistently.

Programming in TeX generally progresses along a very gradual learning curve, requiring a significant investment of time to build custom macros for text formatting. Fortunately, document preparation systems based on TeX, consisting of collections of pre-built macros, do exist. These pre-built macros are time saving, and automate certain repetitive tasks and help reduce user introduced errors; however, this convenience comes at the cost of complete design flexibility. One of the most popular macro packages is called **LaTeX**.

1.2 LaTeX

LaTeX (pronounced either ”Lah-tech” or ”Lay-tech”) is a macro package based on TeX created by Leslie Lamport.Many later authors have contributed extensions, called packages or styles, to LaTeX. Some of these are bundled with most TeX/LaTeX software distributions; more can be found in the Comprehensive TeX Archive Network (CTAN). It is based on the WYSIWYM (what you see is what you mean) idea, meaning you only have focus on the contents of your document and the computer will take care of the formatting. Instead of spacing out text on a page to control formatting, as with Microsoft Word or LibreOffice Writer, users can enter plain text and let LaTeX take care of the rest.

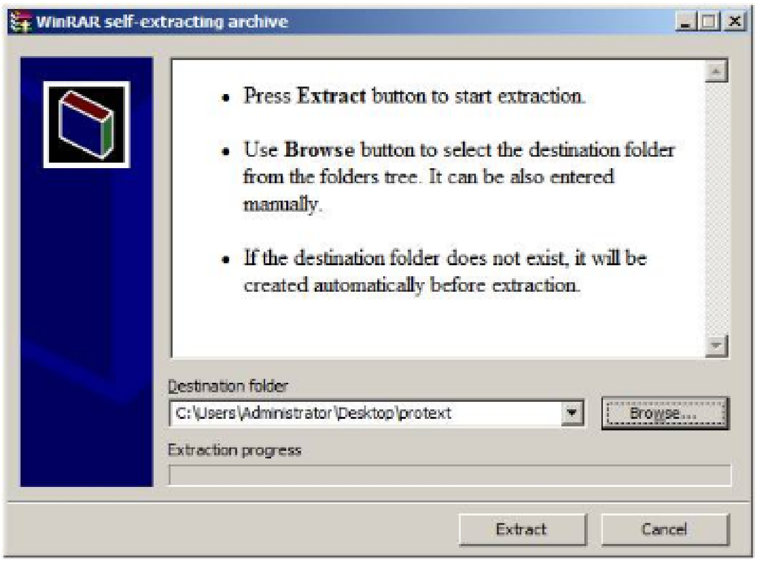
LaTeX is used all over the world for scientific documents, books, as well as many other forms of publishing. Not only can it create beautiful typesetted documents, but it allows users to very quickly tackle the more complicated parts of typesetting, such as inputting mathematics, creating tables of contents, referencing and creating bibliographies, and having a consistent layout across all sections. Due to the huge number of open source packages available the possibilities with LaTeX are endless. These packages allow users to do even more with LaTeX, such as add footnotes, draw schematics, create tables etc.

One of the most important reasons people use LaTeX is that it separates the content of the document from the style. This means that once you have written the content of your document, we can change its appearance with ease. Similarly, you can create one style of document which can be used to standardise the appearance of many different documents. This allows scientific journals to create templates for submissions. These templates have a pre-made layout meaning that only the content needs to be added. In fact there are hundreds of templates available for everything from CVs to slideshows.

1.3 Installing LaTeX on Windows

Prepare to set aside at least an hour of your time to install LaTeX. You should also be on campus or using a high-speed internet connection, since you will have to download a large file. You need to install two different parts, a LaTeX compiler (MiKTeX) and an editor (TeXstudio).

1. Visit `http://mirror.ctan.org/systems/windows/protext/` and click on the **protext.exe** file to download it. This is the proTeXt installer, and it's quite large (1.5 GB), so be prepared to wait a bit while it downloads. (Note: disregard the **Last modified** date; the latest version will be here at any given time.)
2. Once the file has downloaded, double-click on protext.exe. If you see a security warning, click Run.
3. Extract the installation files to your desktop:
   - In the file extracting window, click the **Browse** button. In the Browse for Folder window, click on **Desktop** and then click the **Make New Folder** button. Name the new folder **protext**, click on the new folder to select it, and then click **OK** to close the Browse for Folder window.
   - In the file extracting window, the destination folder should now be set to the protext folder on your Desktop.* Click Extract.

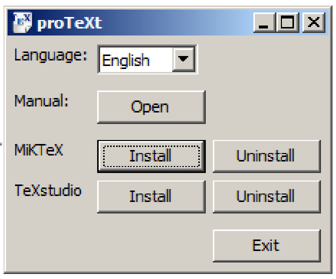

WinRAR self-extracting archive

- Press **Extract** button to start extraction.
- Use **Browse** button to select the destination folder from the folders tree. It can be also entered manually.
- If the destination folder does not exist, it will be created automatically before extraction.

Destination folder: C:\Users\Administrator\Desktop\protext

* It's important to extract the files to a new folder on your Desktop because about 20 files will be extracted!

   - Once the files have been extracted, the window will close.
4. Go to your desktop and then double-click on the protext folder to open it. Double-click on Setup.exe to begin the installation.
5. In the proTeXt pop-up window, click the Install button next to MiKTeX.
   - Read and accept the conditions by checking the box, then click Next.



- Choose to install Basic MikTeX, then click Next.
- Accept the defaults on the next 2 screens by clicking Next.
- In the Settings, choose Letter as the preferred paper size. Make sure ‘Ask Me First’ is chosen for the package installation option. Click Next.
- Click Start to begin the installation. When it is complete, click Next and then Close.

6. In the proTeXt pop-up window, click the Install button next to TeXstudio.
   - Click through the installer, leaving all the defaults.
   - Click Finish when the installer is complete.
7. You have now installed both LaTeX and the editor. You can now delete the downloaded protext.exe file as well as the protext folder on your desktop.

1.4 Installing LaTeX on Linux Machines

There are number of LaTeX distributions you can install on Ubuntu. One such distribution is TeX Live.

1. OPEN YOUR TERMINAL: A terminal is a Command Line Interface (CLI) where you type commands to tell the computer what to do. Make sure you've opened the terminal, if so, continues in the next step.
2. INSTALL TEX LIVE: TeX Live is a TeX distribution to get up and running with the TeX document production system. To install it, once you're in the terminal, enter the following command:

```
sudo apt-get install texlive-full
```

3. INSTALL TEXSTUDIO: Now you need a text editor. I recommend using a specific editor for LaTeX. There are many text editors for LaTeX on the Internet as Kile, TeXworks, JLatexEditor, Gedit LaTeX Plugin, etc. My favorite text editor for Latex is Texstudio. Texstudio is a cross-platform open source LaTeX editor. To install it, go to the Ubuntu or Debian terminal and enter the following command:

```
sudo apt-get install texstudio
```

4. CREATE YOUR FIRST DOCUMENT: To check that everything is working properly, create a LaTeX blank document. Open Texstudio and click on File, New. Then write the following code:

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

Now save the document as a ’tex’ file going to File, Save. Finally, compile the document clicking on Tools, PDFLaTeX. Make sure the ’pdf’ file has been created and it's working. And that's it! You've created your first LaTeX document!

1.5 Spaces

The LaTeXcompiler normalises whitespace so that whitespace characters, such as [space] or [tab], are treated uniformly as a single ”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:

■ **Example 1.1** `It does not matter whether you`
```
enter one or several        spaces
after a word.
```
■

1.6 Reserved Character

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{}
```

The backslash character \ cannot be entered by adding another backslash in front of it (\\); this sequence is used for line breaking. For introducing a backslash in math mode, you can use `\backslash` instead.
The commands \ and \^ produce respectively a tilde and a hat which is placed over the next letter. For example `$\tilde{n}$` gives $\tilde{n}$. You can also use `\textasciitilde` and `\textasciicircum` to enter these characters; or other commands.
If you want to insert text that might contain several particular symbols (such as URIs), you can consider using the `\verb` command.
The ’less than’ (<) and ’greater than’ (>) characters are the only visible ASCII characters (not reserved) that will not print correctly.

1.7 LaTeX Environments

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

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

Between the `\begin` and the `\end` you can put other commands and nested environments.

1.8 Global Structure

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

```
\documentclass{...}
\begin{document}
        ...
\end{document}
```

The area between `\documentclass{...}` and `\begin{document}` is called the ***preamble***. It normally contains commands that affect the entire document.
After the preamble, the text of your document is enclosed between two commands which identify the beginning and end of the actual document:

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

We can put our text where the dots are. The reason for marking off the beginning of our text is that LaTeX allows us to insert extra setup specifications before it . The reason for marking off the end of your text is to provide a place for LaTeX to be programmed to do extra stuff automatically at the end of the document, like making an index.

1.8.1 Document Class

When processing an input file, LaTeX needs to know the type of document the author wants to create. This is specified with the `\documentclass` command. It is recommended to put this declaration at the very beginning.
`\documentclass[options]{class}`
Here, class specifies the type of document to be created. The LaTeX distribution provides additional classes for other documents, including letters and slides. It is also possible to create your own, as is often done by journal publishers, who simply provide you with their own class file, which tells LaTeX how to format your content. But we'll be happy with the standard article class for now. The options parameter customizes the behavior of the document class. The options have to be separated by commas. Example: an input file for a LaTeX document could start with the line
`\documentclass[11pt,twoside,a4paper]{article}`
which instructs LaTeX to typeset the document as an article with a base font size of 11 points, and to produce a layout suitable for double sided printing on A4 paper. Table 1.1 summerizes the major document classes available in LaTeX.

The standard document classes that are a part of LaTeX are built to be fairly generic, which is why they have a lot of options in common. Other classes may have different options (or none at all). Normally, third party classes come with some documentation to let you know. The most common options for the standard document classes are listed in table 1.2

1.8.2 Packages

While writing your document, we may require additional features which basic LaTeX might not be enough. If we want to include graphics, colored text or source code from a file into our document, we need to enhance the capabilities of LaTeX. Such enhancements are called packages. Some packages come with the LaTeX base distribution.Others are provided separately. Modern TeX distributions come with a large number of packages pre-installed. The command to use a package is pretty simple: `\usepackage` :

```
        \usepackage[options]{package}
```

Document Classes	
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.
report	For longer reports containing several chapters, small books, thesis, ...
book	For real 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 writing presentations

Table 1.1: Important **document classes** used in latex

In this command `package` is the name of the package and options is a list of keywords that trigger special features in the package.
For example `circuitikz` package lets us to draw circuits and networks.

1.8.3 Macros

Latex typesetting is made by using special tags or commands that provide a handful of ways to format your document. Sometimes standard commands are not enough to fulfil some specific needs, in such cases new commands can be defined.

LaTeXis shipped with a huge amount of commands for a large number of tasks, nevertheless sometimes is necessary to define some special commands to simplify repetitive and/or complex formatting.

New commands are defined by `\newcommand` statement, let's see an example of the simplest usage.

```
\newcommand{\R}{\mathbb{R}}

The set of real numbers are usually represented
by a blackboard bold capital r: \( \R \).
```

The set of real numbers are usually represented by a blackboard bold capital r: $\mathbb{R}$.

The statement `\newcommand{\R}{\mathbb{R}}` has two parameters that define a new command

`\R`
This is the name of the new command.

`\mathbb{R}`
This is what the new command does. In this case the letter R will be written in blackboard boldface

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 legal- paper can be specified.
`fleqn`	Typesets displayed formulas left-aligned instead of centered.
`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.

Table 1.2: Document Class options

style. (to use *mathbb* is needed the package *amssymb*)

After the command definition you can see how the command is used in the text. Even tough in this example the new command is defined right before the paragraph where it's used, is a good practice put all your user-defined commands in the preamble of your document.

Commands with parameters

It is also possible to create new commands that accept some parameters.

```
\newcommand{\bb}[1]{\mathbb{#1}}

Other numerical systems have similar notations.
The complex numbers \( \bb{C} \), the rational
numbers \( \bb{Q} \) and the integer numbers \( \bb{Z} \).
```

> Other numerical systems have similar notations. The complex numbers $\mathbb{C}$, the rational numbers $\mathbb{Q}$ and the integer numbers $\mathbb{Z}$.

The line `\newcommand{\bb}[1]{\mathbb{#1}}` defines a new command that takes one parameter.

`\bb`
This is the name of the new command.

`[1]`
The number of parameters the new command will take.

`\mathbb{#1}`
This is what the command actually does. In this case the parameter, referenced as `#1`, will be written using blackboard boldface characters. If the defined new command needs more than one parameter, you can refer each parameter by `#1`, `#2` and so on.

1.9 Creating First LaTeX Document

The first step is to create a new LaTeX project. You can do this on your own computer by creating a new .tex file. Always save the tex file inside a folder named after your project. This is a good practice because LaTeXwill create supporting file along with the pdfs and the tex document.

- To Create New File[1]: File -> New
  Save the file in folder specifically meant for the new project file.
- To Begin with,the preamble is created:

  ```
  \documentclass[<document class options>]{<document class>}
  \usepackage{<additional packages>}
  ```

- Next the document environment follows

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

[1]Creation of latex document in this entire book is done with *Tex studio* software. Any other LaTeXeditor software will run the commands used in this without any hassle, but the editor environment might differ.

1.9.1 Top matter

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

A simple example:

```
\documentclass[12pt, a4paper]{article}
\begin{document}
        \title{Raspberry Pi}
        \author{EMpiRIA}
        \date{January 2016}
        \maketitle
\end{document}
```

The `\title`, `\author`, and `\date` commands are self-explanatory. we need to put the title, author name, and date in curly braces after the relevant command. The title and author are usually compulsory (at least if we want LaTeX to write the title automatically); if we omit the `\date` command, LaTeX uses today's date by default. We always 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 we omit `\maketitle`, the titling will never be typeset (unless we write our own).

A more complicated example:

```
\documentclass[12pt, a4paper]{article}
\begin{document}
        \title{Raspberry Pi}
        \author{EMpiRIA\\
        ECE Dept,\\
        AMCEC,\\
        Bangalore.\\
        }
        \date{January 2016}

        \maketitle
\end{document}
```

The double backslash (
) is the LaTeX command for forced linebreaks in tabular material.
if there are two author then we can seperate them using `\and` command:

```
\title{Rapsberry Pi}
\author{Silica \and EMpiRIA}
\date{\today}
\maketitle
```

1.10 Page size and margins

The page dimensions in a LaTeX document are highly configurable and the **geometry** package offers a simple way to change the length and layout of different elements such as the paper size, margins, footnote, header, orientation, etc.

Suppose you have to create a document in a4paper and the text shouldn't exceed 6 in width and 8 in height. To create it with geometry is easy, include this one line in the preamble:

```
\usepackage[a4paper, total={6in, 8in}]{geometry}
```

The parameters passed to the command determine the layout. In this case a4paper establishes the paper size and the total parameter determines the size of the text area.

1.10.1 Paper size, orientation and margins

Paper size, orientation and margins are the most common page elements that must be changed depending on the type of document. To set the desired values there are two ways, either you pass them as parameters to the `\includepackage` statement as in the example above, or use a `\geometry` command in the preamble.

For example, let's create a document with legal paper size, landscape orientation and a 2 in margin:

```
\usepackage[letterpaper, landscape, margin=2in]{geometry}
```

Also, you can achieve the same in a slightly different way

```
\usepackage{geometry}
\geometry{legalpaper, landscape, margin=2in}
```

As you see, the parameters are comma separated. For a complete list of predefined paper sizes see the reference guide. The second parameter is the orientation, its default value is portrait. Finally, each margin is set to 2 in.

1.10.2 Fine tuning your LATEX page dimensions

The regular LATEX page dimensions are presented (with example values) in picture at the right of this section.

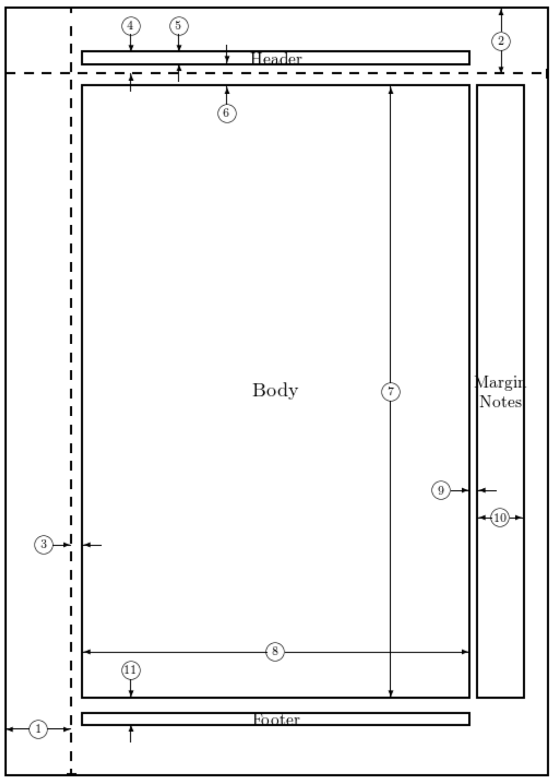
The **geometry** package provides a flexible and easy interface to change page dimensions. You can change the page layout with intuitive parameters:

Next is a list of document elements whose length can be changed. The parameters have to be written in the form parameter=value, use standard LATEX units. (mm, cm, pt, in)

textwidth
Corresponds to element 8 in the figure

textheight
Element 7 in the figure.

total
Depends on other parameters, by default defines the dimensions of the Body, but can be combined



```
1   one inch + \hoffset        2   one inch + \voffset
3   \oddsidemargin = 13pt      4   \topmargin = -23pt
5   \headheight = 12pt         6   \headsep = 25pt
7   \textheight = 674pt        8   \textwidth = 426pt
9   \marginparsep = 10pt       10  \marginparwidth = 50pt
11  \footskip = 30pt               \marginparpush = 5pt (not shown)
    \hoffset = 0pt                 \voffset = 0pt
    \paperwidth = 597pt            \paperheight = 845pt
```

with the includehead, includefoot, includeheadfoot and includemp commands to change the dimensions of Header, the Body, the Footer and the Margin Notes altogether.

left, lmargin, inner
These three parameters change the length of the left margin. Elements 1 and 3 in the figure, combined.

right, rmargin, outer
These three parameters change the length of the right margin. Elements 9 and 10 in the figure, combined.

top, tmargin
These two parameters represent elements 2 and 6 in the figure, combined.

bottom, bmargin
These two parameters set the distance from the bottom edge of the document to its baseline.

headheight
Height of the header

footsep
Separation between the bottom of the text (baseline) and the top of the footnote. Element 11 in the

figure.

footskip
Distance between the baseline of the text and the baseline of the footnote.

marginparwidth, marginpar
Width of the margin notes. Element 10 in the figure.

The *paper size* can be set to any size you need by means of the command `papersize={<width>,<height>}`.

Let's see an example with some of the aforementioned options:

```
\usepackage{geometry}
\geometry{
a4paper,
total={170mm,257mm},
left=20mm,
top=20mm,
}
```

Here the text area, the left margin and the top margin are set. The right and bottom margins are automatically computed to fit the page.

Reference guide

parameter	description	values
papersize	Determines the size of the paper	a0paper, a1paper, a2paper, a3paper, a4paper, a5paper, a6paper, b0paper, b1paper, b2paper, b3paper, b4paper, b5paper, b6paper, c0paper, c1paper, c2paper, c3paper, c4paper, c5paper, c6paper, b0j, b1j, b2j, b3j, b4j, b5j, b6j, ansiapaper, ansibpaper, ansicpaper, ansidpaper, ansiepaper, letterpaper, executivepaper, legalpaper

1.11 Word Formatting

1.11.1 Change of Font Style

To change the fond typeface of the entire document, a simple line must be added to the preamble:

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\usepackage{tgbonum}

\begin{document}
This document is a sample document to test font
families and font typefaces.

This text uses a different font typeface
\end{document}
```

The line \usepackage{tgbonum} establishes the font family *TEXGyreBonum*, whose font package name is tgbonum, as the default font for this document.

The font can also be changed for a specific element in the document.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{geometry}
\geometry{textwidth=7cm}

\usepackage{tgbonum}

\begin{document}
This document is a sample document to
test font families and font typefaces.

{\fontfamily{qcr}\selectfont
This text uses a different font typeface
}
\end{document}
```

The command `\fontfamily{qcr}\selectfont` will set the TEX gyre cursor font typeface, whose fontcode is qcr, for the text inside the braces. A lot more LATEX font typefaces are available, see the reference guide.

The popular LATEX font typefaces are originated from four families:

- *Computer Modern* (default in standard LATEX classes): CM Roman, CM Sans Serif, CM Typewriter
- *Latin Modern*: LM Roman, LM Sans Serif, LM Typewriter, LM Dunhill
- *Post Script Fonts*: Times, Utopia/Fourier, Palatino, Bookman, Helvetica, Courier
- *TEX Gyre*

1.11.2 Font Sizes

LaTeXnormally chooses the appropriate font and font size based on the logical structure of the document (e.g. sections). In some cases, you may want to set fonts and sizes by hand.

Font sizes are identified by special names, the actual size is not absolute but relative to the font size declared in the `\documentclass` statement.

```
In this example the {\huge huge font size} is set and
the {\footnotesize Foot note size also}. There's a fairly
large set of font sizes.
```

In the example, \huge huge font size declares that the text inside the braces must be formatted in a huge font size. For a complete list of available font sizes see the refer the table below.

Command	**Output**
`\tiny`	EMpiRIA
`\scriptsize`	EMpiRIA
`\footnotesize`	EMpiRIA
`\small`	EMpiRIA
`\normalsize`	EMpiRIA
`\large`	EMpiRIA
`\Large`	EMpiRIA
`\LARGE`	EMpiRIA
`\huge`	EMpiRIA
`\Huge`	EMpiRIA

1.11.3 Bold, italics and underlining

Simple text formatting helps to highlight important concepts within a document and make it more readable. Using italics, bold or underlined words can change the perception of the reader.

```
Some of the \textbf{greatest}
discoveries in \underline{science}
were made by \textbf{\textit{accident}}.
```

As you can see in above example, there are three basic commands and they can be nested to get combined effects.

Note: The commands `\it` and `\bf` also work to italicize and boldface text, but it's not recommended to use them since they don't preserve previous styles. With these you can't, for instance, italicize and make a text bold at the same time.

- **Italicized text:**
  To make a text italic is straightforward, use the `\textit` command:

  ```
  Some of the greatest discoveries in science were made by \textit{accident}.
  ```

- **Bold text:**
  To make a text bold use `\textbf` command:

  ```
  Some of the \textbf{greatest} discoveries in science were made by accident.
  ```

- **Underlined text:**
  Underlining text is very simple too, use the `\underline` command:

  ```
  Some of the greatest discoveries in \underline{science} were made by accident.
  ```

- **Emphasising text:**
  Text can be emphasized by using `\emph` command. Sometimes the `\emph` command behaves just as `\textit`, but is not exactly the same:

```
Some of the greatest \emph{discoveries}
in science
were made by accident.

\textit{Some of the greatest \emph{discoveries}
in science
were made by accident.}

\textbf{Some of the greatest \emph{discoveries}
in science
were made by accident.}
```

What the `\emph` command actually does with its argument depends on the context - inside normal text the emphasized text is italicized, but this behaviour is reversed if used inside an italicized text- see example above:

1.12 Using colours in LaTeX

There are several elements in LaTeX whose colour can be changed to improve the appearance of the document. Colours can be manually defined to a desired tone using several models, this article explains how.

The simplest manner to use colours in your LATEX document is by importing the package color or xcolor. Both packages provide a common set of commands for colour manipulation, but the latter is more flexible and supports a larger number of colour models. Below an example:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\usepackage{color}

\begin{document}

This example shows different examples on how to use the \texttt{color} package
to change the colour of elements in \LaTeX.

\begin{itemize}
\color{blue}
\item First item
\item Second item
\end{itemize}

\noindent
{\color{red} \rule{\linewidth}{0.5mm} }

\end{document}
```

This example shows different examples on how to use the `color` package to change the colour of elements in LaTeX.

- First item
- Second item

Note: In all the examples the package xcolor can be used instead of color

In this example, the package color is imported with

`\usepackage{color}` then the command `\color{blue}` sets the blue colour for the current block of text. In this case for the itemize environment.

The colour of a second block of text, delimited by { and }, is set to red with the command `\color{red}`, then a 0.5mm-thick horizontal ruler is inserted by `\rule{\linewidth}{0.5mm}`.

The amount of available colour names depends on the driver, usually the next colours can be used with any driver: white, black, yellow, green, blue, purple cyan and magenta.

1.12.1 Basic usage

The colour system provided by the packages color and xcolor is built around the idea of colour models, the colour mode and the colour names supported by a driver vary.

The model based on colour names is very intuitive, even though the list of available names is limited, usually provides enough options. Below an example:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\usepackage[usenames, dvipsnames]{color}

\begin{document}

This example shows different examples on how to use the \texttt{color} package
to change the colour of elements in \LaTeX.

\begin{itemize}
\color{ForestGreen}
\item First item
\item Second item
\end{itemize}

\noindent
{\color{RubineRed} \rule{\linewidth}{0.5mm} }

The background colour of some text can also be \textcolor{red}{easily} set. For
instance, you can change to orange the background of \colorbox{BurntOrange}{this
text} and then continue typing.

\end{document}
```

There are a few changes in this example compared to the one presented in the introduction. First, the command to import the color package has two additional parameters:

- usenames Makes the names in the corresponding driver name model available. This option can be omitted in xcolor.

- dvipsnames Makes the colour names for the driver dvips available, if the package color is imported, this option must be used in conjunction with usenames. From this new set of colour names, the example uses: *ForestGreen*, *RubineRed and BurntOrange*.

Other possible drivers are: xdvi, dvipdf, pdftex, dvipsone, dviwin, emtex, truetex and xtex.

Two new commands are also presented in the example:

```
\textcolor{red}{easily}
```

Changes the colour of inline text. Takes two parameters, the colour to use and the text whose colour is changed. In the example the word easily is printed in red.

```
\colorbox{BurntOrange}{this text}
```

Changes the background colour of the text passed as second parameter. In the example the words this text are printed in BurntOrange.

1.12.2 Creating your own colours

It is possible to define your own colours, the manner in which the colour is defined depends on the preferred model. Below an example using the 4 colour models typically supported by any driver.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\usepackage[usenames, dvipsnames]{color}

\definecolor{mypink1}{rgb}{0.858, 0.188, 0.478}
\definecolor{mypink2}{RGB}{219, 48, 122}
\definecolor{mypink3}{cmyk}{0, 0.7808, 0.4429, 0.1412}
\definecolor{mygray}{gray}{0.6}

\begin{document}
User-defined colours with different colour models:

\begin{enumerate}
\item \textcolor{mypink1}{Pink with rgb}
\item \textcolor{mypink2}{Pink with RGB}
\item \textcolor{mypink3}{Pink with cmyk}
\item \textcolor{mygray}{Gray with gray}
\end{enumerate}

\end{document}
```

The command `\definecolor` takes three parameters: the name of the new colour, the model, and the colour definition. Roughly speaking, each number represent how much of each colour you add to the mix that makes up the final colour.

- **rgb:** Red, Green, Blue. Three comma-separated values between 0 and 1 define the components of the colour.
- **RGB:** The same as rgb, but the numbers are integers between 0 and 255.
- **cmyk:** Cyan, Magenta, Yellow and blacK. Comma-separated list of four numbers between 0 and 1 that determine the colour according to the additive model used in most printers.

- **gray:** Grey scale. A single number between 0 and 1.

In the example, mypink1, mypink2 and mypink3 define the same colour but for different models. You can actually see that the one defined by cmyk is slightly different.

Colours defined by either model can later be used within your document not only to set the colour of the text, but for any other element that takes a colour as parameter, for instance tables (you must add the parameter table to the preamble), graphic elements created with TikZ, plots, vertical rulers in multicolumn documents and code listings.

1.12.3 xcolor-only colour models

There are some additional commands that are only available with the package xcolor, these enable support for more colour models and friendly colour mixing.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\usepackage[dvipsnames]{xcolor}
\colorlet{LightRubineRed}{RubineRed!70!}
\colorlet{Mycolor1}{green!10!orange!90!}
\definecolor{Mycolor2}{HTML}{00F9DE}

\begin{document}

This document present several examples on how to use the \texttt{color} package
to change the colour of elements in \LaTeX.

\begin{itemize}
\item \textcolor{Mycolor1}{First item}
\item \textcolor{Mycolor2}{Second item}
\end{itemize}

\noindent
{\color{LightRubineRed} \rule{\linewidth}{1mm} }

\noindent
{\color{RubineRed} \rule{\linewidth}{1mm} }
```

Three new colours are defined in this example, each one in a different manner.

- `\colorlet{LightRubineRed}{RubineRed!70!}`:
  A new colour named LightRubineRed is created, this colour has 70% the intensity of the original RubineRed colour. You can think of it as a mixture of 70% RubineRed and 30% white. Defining colours in this way is great to obtain different tones of a main colour, common practice in corporate brands. In the example, you can see the original RubineRed and the new LightRubineRed used in two consecutive horizontal rulers.
- `\colorlet{Mycolor1}{green!10!orange!90!}`:
  A colour named Mycolor1 is created with 10% green and 90% orange. You can use any number of colours to create new ones with this syntax.

- `\definecolor{Mycolor2}{HTML}{00F9DE}`:
  The colour Mycolor2 is created using the HTML model. Colours in this model must be created with 6 hexadecimal digits, the characters A,B,C,D,E and F must be upper-case.

  The colour models that only xcolor support are:

  – **cmy** cyan, magenta, yellow
  – **hsb** hue, saturation, brightness
  – **HTML** RRGGBB
  – **Gray** Grey scale, a number between 1 and 15.
  – **wave** Wave length. Between 363 and 814

1.13 Text Alignment

1.13.1 Left-justified text

The default environment for left-alignment is flushleft

```
\begin{flushleft}
This is left justified.
\end{flushleft}

The Corresponding LaTeX output is:
```

This is left justified.

All the text in between `\begin{flushleft}` and `\end{flushleft}` is left-justified.

1.13.2 Right-justified text

Right-aligning text is straightforward with the environment `\flushright`.

```
\begin{flushright}
This is right justified.
\end{flushright}

The Corresponding LaTeX output is:
```

This is right justified.

Text in between `\begin{flushright}` and `\end{flushright}` is right-justified.

1.13.3 Centered text

To centre a block of text use the environment `\center`.

```
\begin{center}
This is centered.
\end{center}

The Corresponding LaTeX output is:
```

This is centered.

Text in between `\begin{center}` and `\end{center}` is centred. The switch command `\centering` will also produce centred text, but the behaviour is different; in this case the text will be centred from the point where the command is declared till another switch command is used. This is more suitable for large blocks of text or for the whole document.

1.13.4 Fully-justified text

To justify a block of text use the environment `\justify`.

```
\justify Hello, here is some text without a meaning.  This text should show what
a printed text will look like at this place.  If you read this text,
you will get no information.
```

1.14 Paragraph formatting

The default LATEXformatting is fine and makes documents quite readable, but it can be changed if you need a different looking document. This article explains how to change the paragraph and line spacing.

Changing the length of some specific elements my alter the looking of the entire document by adding these lines in the preamble.

```
 \setlength{\parindent}{4em}
 \setlength{\parskip}{1em}
 \renewcommand{\baselinestretch}{2.0}
```

To start a new paragraph in LATEX, as said before, you must leave a blank line in between. There's another way to start a new paragraph, look at the following snippet.

```
This is the text in first paragraph. This is the text in first
paragraph. This is the text in first paragraph. \par
This is the text in second paragraph. This is the text in second
paragraph. This is the text in second paragraph.
```

This is the text in first paragraph. This is the text in first paragraph. This is the text in first paragraph.
    This is the text in second paragraph. This is the text in second paragraph. This is the text in second paragraph.

As you can see, the `\par` command starts a new paragraph without the need of a blank line.

1.14.1 Paragraph Indentation

By default, LATEXdoes not indent the first paragraph of a section or a chapter. The size of the subsequent paragraph indents is determined by `\parindent`

```
\setlength{\parindent}{4em}

\begin{document}

This is the text in first paragraph. This is the text in first
paragraph. This is the text in first paragraph. \par
This is the text in second paragraph. This is the text in second
paragraph. This is the text in second paragraph.

\end{document}
```

The default length of this parameter is set by the document class used. It is possible to change the indent size. In the example, the first lines of each paragraph are indented 4em (an "em" equals the length of the "m" in the current font), this is accomplished by the command `\setlength{\parindent}{4em}`. It's recommended to put this command in the preamble of the document, but it can be set anywhere else.

If you want to create a non-indented paragraph, like the second one in the example, put the command `\noindent` at the beginning of it. If you want the whole document not to be indented, set the indentation length to zero with `\setlength{\parindent}{0pt}`.

On the other side, if you want to indent a paragraph that is not indented you can use `\indent` right above it. It should be noted that this command will only have an effect when `\parindent` is set to zero.

1.14.2 Paragraph spacing

The length parameter that characterises the paragraph spacing is `\parskip`, this determines the space between a paragraph and the preceding text.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\setlength{\parindent}{4em}
\setlength{\parskip}{1em}

\begin{document}
This is the text in first paragraph. This is the text in first
```

```
paragraph. This is the text in first paragraph. \par
This is the text in second paragraph...

\end{document}
```

In the example, the command `\setlength{\parskip}{1em}` sets the paragraph separation to 1em.

1.14.3 Line Spacing

There are three commands that control the line spacing, below an example redefining the length of `\baselinestretch`.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\setlength{\parindent}{4em}
\setlength{\parskip}{1em}
\renewcommand{\baselinestretch}{1.5}

\begin{document}
This is the text in first paragraph. This is the text in first
paragraph. This is the text in first paragraph. \par
This is the text in second paragraph...
\end{document}
```

In the example above, `\renewcommand{\baselinestretch}{1.5}` scales the default interline space to 1.5 its default value. Of course that number can be set to any value.

As mentioned before, there are other two LATEX lengths that may change the line spacing:

`\baselineskip`
Is a length determining the minimum space between the bottom of two successive lines in a paragraph; it may be changed (in the preamble) by `\setlength{\baselineskip}{value}`. Where value is set using any of the LaTeX units.

`\linespread{value}`
where value determine line spacing. This value is somewhat confusing, because:

Value	**Line Spacing**
1.0	single spacing
1.3	one-and-a-half spacing
1.6	double spacing

1.15 Lists

1.15.1 Unordered List

The unordered (unnumbered) lists are produced by the itemize environment. Each entry must be preceded by the control sequence `\item`.

```
\begin{itemize}
\item The individual entries are indicated with a black dot, a so-called bullet.
\item The text in the entries may be of any length.
\end{itemize}
```

- The individual entries are indicated with a black dot, a so-called bullet.
- The text in the entries may be of any length.

By default the individual entries are indicated with a black dot, so-called bullet. The text in the entries may be of any length.

1.15.2 Ordered List

Ordered list have the same syntax inside a different environment:

```
\begin{enumerate}
\item The labels consists of sequential numbers.
\item The numbers starts at 1 with every call to the enumerate environment.
\end{enumerate}
```

1. The labels consists of sequential numbers.
2. The numbers starts at 1 with every call to the enumerate environment.

The ordered lists are generated by a `\enumerate` environment and each entry must be preceded by the control sequence `\item`, which will automatically generate the number labelling the item. The enumerate labels consists of sequential numbers, these numbers starts at 1 with every call to the enumerate environment.

1.15.3 Nested List

In LaTeX you can insert a list inside another list. The above lists may be included within one another, either mixed or of one type, to a depth of four levels.

```
\begin{enumerate}
\item The labels consists of sequential numbers.
\begin{itemize}
\item The individual entries are indicated with a black dot, a so-called bullet.
\item The text in the entries may be of any length.
\end{itemize}
\item The numbers starts at 1 with every call to the enumerate environment.
\end{enumerate}
```

1. The labels consists of sequential numbers.
   - The individual entries are indicated with a black dot, a so-called bullet.
   - The text in the entries may be of any length.
2. The numbers starts at 1 with every call to the enumerate environment.

1.16 Headers and Footers

LaTeXhas some predetermined styles that change the way the header and the footer are displayed. The footer and the header can also be customized to fit any particular layout. This article explains how.

The information displayed in the footer and the header of a document depends on the page style currently active, these page styles are more notorious in the book document class:

```
\documentclass[a4paper,12pt,twoside]{book}
\usepackage[english]{babel}
\usepackage[utf8]{inputenc}

\pagestyle{headings}

\begin{document}
\chapter{Sample Chapter}
\section{New section}

Hello, here is some text without a meaning.  This text should
show what aprinted text will look like at this place.  If you
read this text, you will get noinformation.  Really?  Is there
no information?  Is there a dence betweenthis text and some
nonsense like \Huardest gefburn"?  Kjift { not at all!...

\end{document}
```

The command `\pagestyle{headings}` sets the page style called headings to the current document.

1.16.1 Standard page styles

The standard page styles are invoked in LATEX by means of the command:

```
\pagestyle{''style''}
\pagestyle{myheadings}
```

The myheadings pagestyle displays the page number on top of the page in the outer corner.

There are other three page styles:

- `empty`: Both the header and footer are cleared (blank) in this page style.
- `plain`: This is the default style. The header is empty and the footer contains page numbers in the centre.

- `myheadings`: The footer is empty in this page style. The header contains the page number on right side (on even pages) or on left side (on odd pages) along with other user-supplied information; there is an exception for the first page of each chapter, where the footer contains centred page number while the header is blank.

1.16.2 Setting page style for current page only

Sometimes is convenient to specify the page style only for the current page. For instance, to leave a intentionally blank page or to remove the header and footer from the current chapter page:

```
\chapter{Sample Chapter}
\thispagestyle{empty}

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim
}ad minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit es...
\end{document}
```

1.16.3 Style customization in single-sided documents

Styles can be modified beyond the standard layouts by means of **fancyhdr**. Below is an example.

```
\documentclass{article}
\usepackage[english]{babel}
\usepackage[utf8]{inputenc}
\usepackage{fancyhdr}

\pagestyle{fancy}
\fancyhf{}
\rhead{EMpiRIA}
\lhead{Guides and tutorials}
\rfoot{Page \thepage}

\begin{document}

\section{First Section}

Hello,  here  is  some  text  without  a  meaning.   This
text  should  show  what  aprinted text will look like at
this place.  If you read this text, you will get noinformation.
Really?  Is there no information?  Is there a dence between
this ...

\end{document}
```

To customize the footer and header in your document first import the package **fancyhdr** with

```
\usepackage{fancyhdr}
```

After that, the “fancy” style is set by `\pagestyle{fancy}`. The command `\fancyhf{}` clears the header and footer, otherwise the elements of the default “plain” page style will appear.

Below, a description of the rest of the commands and a few more whose usage is similar.

`\rhead{EMpiRIA}`
Prints the text included inside the braces on the right side of the header.

`\lhead{Guides and tutorials}`
Prints the text set inside the braces on the left side of the header.

`\chead{ }`
Similar to the previous commands, in this case the text is centred on the header.

`\rfoot{Page \thepage}`
Prints the word `Page` and next the page number which is automatically set by `\thepage` on the right side of the footer.

`\lfoot{ }`
This prints the parameter passed inside the braces on the left side of the footer.

`\cfoot{ }`
Similar to the previous two commands, prints its parameter on the centre of the footer.

1.16.4 Decorative lines on header and footer

When you are using fancyhdr in your document, there are two decorative lines on both the header and the footer, the latter has 0pt thickness and hence is not visible. It’s easy to change that:

```
\documentclass[a4paper,12pt,twoside]{book}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}
\usepackage{fancyhdr}

\pagestyle{fancy}
\fancyhf{}
\fancyhead[LE,RO]{EMpiRIA}
\fancyhead[RE,LO]{Guides and tutorials}
\fancyfoot[CE,CO]{\leftmark}
\fancyfoot[LE,RO]{\thepage}

\renewcommand{\headrulewidth}{2pt}
\renewcommand{\footrulewidth}{1pt}

\begin{document}

\chapter{Using different page styles}

Lorem ipsum dolor sit amet, consectetur adipiscing ...
```

There are two additional lines in this example:

```
\renewcommand{\headrulewidth}{2pt}
```
This sets the header line thickness to 2pt.
```
\renewcommand{\footrulewidth}{1pt}
```
Sets the footer line thickness to 1pt.

1.17 Page Numbering

Page numbering in LATEX uses Arabic numbers by default, but this can be changed to use Roman numerals and/or letters. You can also combine several numbering styles in a single document.

Setting a numbering style is straightforward

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\pagenumbering{roman}

\begin{document}
\tableofcontents

\section{Testing section}
...

\end{document}
```

The command `\pagenumbering{roman}` will set the page numbering to lowercase Roman numerals.

1.17.1 Numbering styles

There are several numbering styles, at the introduction the lower-case Roman numerals were presented, let's see a second example:

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\pagenumbering{alph}

\begin{document}
\tableofcontents

\section{Testing section}
...

\end{document}
```

As mentioned before, following command is used to set the layout of the page number.

`\pagenumbering{num_style}`

Below, a list of styles available for page numbering:

- `arabic`: arabic numerals
- `roman`: lowercase roman numerals
- `Roman`: uppercase roman numerals
- `alph`: lowercase letters
- `Alph`: uppercase letters

The position where the number appear in the page is determined by the page style used in the document.

1.17.2 Using two page numbering styles in a single document

In books, is customary to use Roman numerals for the pages before the first chapter/section, and Arabic numbers for the rest of the document. There are two commands available in the book document class that accomplish this:

```
\documentclass{book}
\usepackage[utf8]{inputenc}

\begin{document}

\frontmatter

\addcontentsline{toc}{chapter}{Foreword}
\Some text...

\addcontentsline{toc}{chapter}{Dummy entry}
Some text...

\tableofcontents

\mainmatter

\chapter{First Chapter}

This will be an empty chapter...

\end{document}
```

The commands that control the page numbering are:

`\frontmatter`

The pages after this command and before the command `\mainmatter`, will be numbered with lowercase Roman numerals.

`\mainmatter`

This will restart the page counter and change the style to Arabic numbers.

If your document class is not book or you need more control over the page counter and the numbering style, see the next example:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\pagenumbering{roman}

\begin{document}
\tableofcontents

\section{First section}
\setcounter{page}{3}

Some text here...

\section{Second section}
Some more text here..

\section{Heading on Level 1 (section)}
\pagenumbering{arabic}

More text here...
\end{document}
```

In the example above the numbering styles are explicitly established and the counter set to a specific number. Below a description of each command is provided:

`\pagenumbering{roman}`
This command sets the page numbers to lowercase Roman numerals.
`\setcounter{page}{3}`
This will manually set the page counter to 3 in this page, subsequent pages are numbered starting the count from this one.
`\pagenumbering{arabic}`
The page numbering is switched to Arabic, this will also restart the page counter.

1.17.3 Customizing numbering styles

With the aid of the package fancyhdr we can customize how the page numbers are displayed. For example, if you want to put the current page number in the context of the page numbers in the whole document (1 of 120, for instance) you can do that as shown below:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}
\usepackage{fancyhdr}
\usepackage{lastpage}
```

```
\pagestyle{fancy}
\fancyhf{}

\rfoot{Page \thepage \hspace{1pt} of \pageref{LastPage}}

\begin{document}

\tableofcontents

\section{First section}
Some text...

\section{Second section}
More text...

\end{document}
```

Here, the package lastpage is imported by

```
\usepackage{lastpage}
```

so we can refer the last page of the document. Then we set **fancyhdr** to display on the downer right corner the text "Page n of All" where n is the current page and *All* is the last page.

1.18 Footnotes

In some documents it's necessary to put annotations at the bottom of the page. This article explains how to add footnotes.

Adding a footnote to your document is straightforward

```
I'm writing something here to test \footnote{footnotes working fine}
several features.
```

The command `\footnote{footnotes working fine}` adds a superscript to the word right before the command and prints the corresponding footnote.

1.18.1 Basic usage

The superscript mark to reference a footnote can be manually set. See the example below:

```
I'm writing something here to test \footnote[10]{footnotes working fine}
several features. You can write the footnote text\footnotemark in its
own line.
\footnotetext{Second footnote}
```

There are three new commands here:

`\footnote[10]{footnotes working fine}`
Adds a footnote using "10" as reference mark. Unless you have a good reason to do this, it's not recommended because the footnote counter is not altered and you may end up with two different

footnotes with the same mark.

```
\footnotemark
```
Prints a foot note mark but without the actual footnote. This is helpful to write the actual footnote text in a new line.

```
\footnotetext{Second footnote}
```
Prints the footnote corresponding to the previous `\footnotemark`.

1.19 Hyperlinks

Let's start with a minimal working example, by simply importing the hyperref package all cross-referenced elements become hyperlinked.

```
\documentclass{book}
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}

\usepackage{hyperref}

\begin{document}

\frontmatter

\tableofcontents

...
\end{document}
```

The lines in the table of contents become links to the corresponding pages in the document by simply adding in the preamble of the document the line.

```
\usepackage{hyperref}
```

One must be careful when importing hyperref. Usually it has to be the last package to be imported, but there might be some exceptions to this rule.

1.19.1 Linking web addresses

Links to a web address or email can added to a LATEX file using the `\url` command to display the actual link or `\href` to use a hidden link and show a word/sentence instead.

```
For further references see \href{http://www.amceducation.in}{Something Linky}
or go to the next url: \url{http://www.amceducation.in}
```

For further references see Something Linky or go to the next url: `http://www.amceducation.in`

There are two commands in the example that generate a link in the final document:

```
\href{http://www.amceducation.in}{Something Linky}
```

There are two parameters passed to this command, the first one is the url to the link, http://www.amceducation.in in this case, and the second one is the clickable text to be shown, Something Linky.

```
\url{http://www.sharelatex.com}
```

This command will show the url passed as parameter and make it into a link, useful if you will print the document.

1.19.2 Linking local files

The commands `\href` and `\url` presented in the previous section can be used to open local files.

```
For further references see \href{http://www.amceducation.in}{Something Linky}
or go to the next url: \url{http://www.amceducation.in} or open the next
file \href{run:./file.txt}{File.txt}
```

> For further references see Something Linky or go to the next url: `http://www.amceducation.in` or open the next file File.txt

The command `\href{run:./file.txt}{File.txt}` prints the text "File.txt" that links to a local file called "file.txt" located in the current working directory. Notice the text "run:" before the path to the file.

The file path follows the conventions of UNIX systems, using . to refer the current directory and .. for the previous directory.

The command `\url{}` can also be used, with the same syntax described for the path, but it's reported to have some problems.

1.19.3 Inserting links manually

It was mentioned before that all cross-referenced elements become links once hyperref is imported, thus we can use `\label` anywhere in the document and refer later those labels to create links. This is not the only manner to insert hyperlinks manually.

```
It's also possible to link directly any word
or \hyperlink{thesentence}{any sentence} in you document.

If you read this text, you will get no information.  Really?
Is there no information?

For instance \hypertarget{thesentence}{this sentence}.
```

> It's also possible to link directly any word or any sentence in you document.
>
> If you read this text, you will get no information. Really? Is there no information?
>
> For instance this sentence.

There are two commands to create user-defined links.

```
\hypertarget{thesentence}{this sentence}
```

The first parameter passed inside braces to this command is a unique identifier for this sentence. The second parameter is the text "this sentence", and will be printed normally, but when a link pointing to the identifier "thesentence" is clicked the PDF file will scroll to this point.

```
\hyperlink{thesentence}{any sentence}
```

This command prints the text "any sentence" as a clickable element that redirects to the point whose identifier is "thesentence".

2. Tables

Tables are common elements in most scientific documents, LaTeXprovides a large set of tools to customize tables, change the size, combine cells, change the colour of cells and so on. This article explains how.

2.1 Introduction

Below you can see the simplest working example of a table

```
\begin{center}
\begin{tabular}{ c c c }
cell1 & cell2 & cell3 \\
cell4 & cell5 & cell6 \\
cell7 & cell8 & cell9
\end{tabular}
\end{center}
```

For the example shown above, the table with three rows and three columns would be created as shown below.

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

2.2 Creating simple table

The tabular environment is more flexible, we create it by inserting separator lines in between each column.

```
\begin{center}
```

```
\begin{tabular}{ |c|c|c| }
\hline
Sl.no & Subject & Subject code \\
1 & AdHoc & 15EC844 \\
2 & LIC & 15EC46 \\
\hline
\end{tabular}
\end{center}
```

Sl.no	Subject	Subject code
1	AdHoc	15EC844
2	LIC	15EC46

It was already said that the tabular environment is used to type tables. To be more clear about how it works below is a description of each command.

{ |c|c|c| }
This declares that three columns, separated by a vertical line, are going to be used in the table. Each c means that the contents of the column will be centered, you can also use r to align the text to the right and l for left alignment.

`\hline`
This will insert a horizontal line on top of the table and at the bottom too. There is no restriction on the number of times you can use `\hline`.

`Sl.no & Subject & Subject code`
Each & is a cell separator and the double-backslash \\ sets the end of this row.

2.3 Tables with fixed length

When formatting a table you might require a fixed length either for each column or for the entire table. In the example below a fixed column width is established.

First, to use the parameters shown in the example, you must import the package array in the preamble of your LaTeXfile with the next command

```
\usepackage{array}

\begin{center}
\begin{tabular}{ |m{2em}|m{2cm}|m{2cm}| }
\hline
Sl.no & Subject & Subject code  \\
\hline
1 & AdHoc & 15EC844 \\
\hline
2 & LIC & 15EC46 \\
\hline
\end{tabular}
\end{center}
```

Sl.no	Subject	Subject code
1	AdHoc	15EC844
2	LIC	15EC46

In the tabular environment, the parameter m{2em} sets a length of 2em for first column (2cm for the other two) and centres the text in the middle of the cell. The aligning options are m for middle, p for top and b for bottom. In standard tables new lines must be inserted manually so the table won't stretch out of the text area, when using this parameters the text is automatically formatted to fit inside each cell.

2.4 Changing text direction in a cell

This can be done using the `\rotatebox` macro of graphicx. The syntax is:

```
\rotatebox[origin=<O>]{<angle in degree>}{<Text>}
```

The origin on which the text is rotated can be: l (left), r (right), c (center) and t (top), b (bottom), B (baseline). For example rt is the right top corner. Default is lB.

```
\begin{center}
\begin{tabular}{ |m{2em}|m{2cm}|m{2cm}| }
\hline
Sl.no & Subject & Subject code  \\
\hline
1 & AdHoc & 15EC844 \\
\hline
2 & LIC & 15EC46 \\
\hline
\rotatebox{45}{45deg} & \rotatebox{90}{90deg} & \rotatebox[origin=rB]{-45}{135deg}\\
\hline
\end{tabular}
\end{center}
```

Sl.no	Subject	Subject code
1	AdHoc	15EC844
2	LIC	15EC46
45deg	90deg	135deg

2.5 Combining rows and columns

Rows and columns can be combined in a bigger cell. The example below is an example of the `\multicolumn` command to combine columns.

```
\begin{tabular}{ |p{1cm}|c|m{4cm}|p{2cm}|p{2cm}|p{2cm}| }
\hline
```

```
Sl.\newline No &  Subject & \centering Title &
\multicolumn{3}{|c|}{Examination}  \\
\cline{4-6}
& & & Marks& IA marks& Total\\
\hline
& & & & & \\
\hline
\multicolumn{3}{|c|}{Total}& & & \\
\hline
\end{tabular}
```

Sl. No	Subject	Title	Examination		
			Marks	IA marks	Total
Total					

Let's see each part of the command `\multicolumn{3}{|c|}{Examination}\\` :

{3}
The number of columns to be combined, 3 in this case.

{|c|}
Delimiters and alignment of the resulting cell, in this case the text will be centred and a vertical line will be drawn at each side of the cell.

{Examination}
Text to be displayed inside the cell.

To combine rows the package **multirow** must be imported with `\usepackage{multirow}` in your preamble, then you can use the `\multirow` command in your document:

```
\begin{tabular}{ |p{1cm}|c|m{4cm}|p{2cm}|p{2cm}|p{2cm}| }
\hline
\multirow{2}{3em}{Sl.no} &\multirow{2}{3em}{Subject} &\multirow{2}{3em}{Title} &
\multicolumn{3}{|c|}{Examination}  \\
\cline{4-6}
& & & Marks& IA marks& Total\\
\hline
& & & & & \\
\hline
\multicolumn{3}{|c|}{Total}& & & \\
\hline
\end{tabular}
```

Sl.no	Subject	Title	Examination		
			Marks	IA marks	Total
Total					

The command multirow takes three parameters. The first one is the number of rows to be combined, 3 in the example. The second parameter is the width of the column, 3em in the example. Finally, the third parameter is the content of the cell.

2.6 Multi-page tables

To insert a very long table, which takes up two or more pages in your document, use the longtable package. First, add to the preamble the line

```
\usepackage{longtable}
```

This will make the command *longtable* available.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{longtable}

\begin{document}

\begin{longtable}[c]{| c | c |}
\caption{Long table caption.\label{long}}\\

\hline
\multicolumn{2}{| c |}{Begin of Table}\\
\hline
Something & something else\\
\hline
\endfirsthead

\hline
\multicolumn{2}{|c|}{Continuation of Table \ref{long}}\\
\hline
Something & something else\\
\hline
\endhead

\hline
\endfoot

\hline
\multicolumn{2}{| c |}{End of Table}\\
\hline\hline
\endlastfoot
```

```
Lots of lines & like this\\
Lots of lines & like this\\
Lots of lines & like this\\
Lots of lines & like this\\
Lots of lines & like this\\
Lots of lines & like this\\
Lots of lines & like this\\
Lots of lines & like this\\
...
Lots of lines & like this\\
\end{longtable}
\end{document}
```

Table 2.1: Long table caption.

Begin of Table	
Something	something else
Lots of lines	like this
Lots of lines	like this
Lots of lines	like this
Lots of lines	like this
Lots of lines	like this
Lots of lines	like this
Lots of lines	like this
Lots of lines	like this
... Lots of lines	like this
End of Table	

longtable behaviour is similar to the default tabular, but generates tables that can be broken by the standard LaTeXpage-breaking algorithm. There are four elements long-table specific.

`\endfirsthead`
Everything above this command will appear at the beginning of the table, in the first page.

`\endhead`
Whatever you put before this command and below endfirsthead will be displayed at the top of the table in every page except the first one.

`\endfoot`
Similar to `\endhead`, what you put after `\endhead` and before this command will appear at the bottom of the table in every page except the last one.

```
\endlastfoot
```

Similar to endfisthead. The elements after `\endfoot` and before this command will be displayed at the bottom of the table but only in the last page where the table appears.

2.7 Rotating the table

It is important to include the package `rotating` with

```
\usepackage{rotating}
```

in your preamble inorder to use `\begin{sidewaystable}` and `\end{sidewaystable}` to rotate the table.

```
\begin{sidewaystable}
\flushright
\begin{tabular}{ |p{1cm}|c|m{1cm}|p{1cm}|p{1cm}|p{1cm}| }
\hline
Sl.\newline No &  Subject & Title &
\multicolumn{3}{|c|}{Examination}  \\
\cline{4-6}
& & & Marks& IA marks& Total\\
\hline
& & & & & \\
\hline
& & & & & \\
\hline
\end{tabular}
\end{sidewaystable}
```

2.8 Captions, Labels and References

Tables can be captioned, labelled and referenced by means of the table environment.

```
\begin{table}[!h]
\centering
\begin{tabular}{ |m{2em}|m{2cm}|m{2cm}| }
\hline
Sl.no & Subject & Subject code  \\
\hline
1 & AdHoc & 15EC844 \\ [3ex]
\hline
2 & LIC & 15EC46 \\
\hline
\end{tabular}
\caption{Table to test captions and labels}
\label{table:1}
\end{table}
```

Sl.no	Subject	Subject code
1	AdHoc	15EC844
2	LIC	15EC46

Table 2.2: Table to test captions and labels

There are three important commands in the example:

`\caption{Table to test captions and labels}`
As you may expect this command sets the caption for the table, if you create a list of tables this caption will be used there. You can place it above or below the table.

`\label{table:1}`
If you need to refer the table within your document, set a label with this command. The label will number the table, and combined with the next command will allow you to reference it.

`\ref{table:1}`
This code will be substituted by the number corresponding to the referenced table.

The parameter h! passed to the table environment declaration establishes that this table must be placed here, and override LaTeXdefaults. Other positioning parameters can be passed also:

h
Will place the table here approximately.

t
Position the table at the top of the page.

b
Position the table at the bottom of the page.

p
Put the table in a special page, for tables only.

!
Override internal LATEX parameters.

H
Place the table at this precise location, pretty much like h!. For further examples on table positioning see the Positioning images and tables article.

In this example there are a few more commands.:

`\centering`
Centres the table relative to the float container element.

`\[3ex]`
This adds extra space to the cell.

2.9 Changing the appearance of a table

Several table elements can be modified to achieve a good-looking document. Below you will learn how to modify the line thickness, the line colour and the background colour of the cells in your table.

2.9.1 Line width and cell padding

The readability of the table sometimes is improved by incrementing the column spacing and row stretch.

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\setlength{\arrayrulewidth}{1mm}
\setlength{\tabcolsep}{18pt}
\renewcommand{\arraystretch}{1.5}

\begin{document}
\begin{tabular}{ |m{2em}|m{2cm}|m{2cm}| }
\hline
Sl.no & Subject & Subject code  \\
\hline
1 & AdHoc & 15EC844 \\
\hline
2 & LIC & 15EC46 \\
\hline
\end{tabular}
\end{document}
```

A description of the commands is provided below:

`\setlegth{\arrayrulewidth}{1mm}`
This sets the thickness of the borders of the table. In the example is 1mm but you can use other units, see the article Lengths in LaTeX for a complete list.

`\setlength{\tabcolsep}{18pt}`
The space between the text and the left\right border of its containing cell is set to 18pt with this command. Again, you may use other units if needed.

`\renewcommand{\arraystretch}{1.5}`
The height of each row is set to 1.5 relative to its default height.

2.9.2 Colour alternating rows

It is a common practice to use two colours for alternating rows in a tables to improve readability. This can be achieved in LaTeXwith the package xcolor and the table parameter.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[table]{xcolor}

{\rowcolors{4}{green!80!yellow!50}{green!70!yellow!40}
\begin{tabular}{ |p{1cm}|c|m{4cm}|p{2cm}|p{2cm}|p{2cm}| }
\hline
\multirow{2}{3em}{Sl.no} &\multirow{2}{3em}{Subject} &\multirow{2}{3em}{Title} &
\multicolumn{3}{|c|}{\cellcolor{blue!10}Examination}  \\
\cline{4-6}
& & & Marks& IA marks& Total\\
\hline
& & & & & \\
\hline
& & & & & \\
\hline
& & & & & \\
\hline
\multicolumn{3}{|c|}{Total}& & & \\
\hline
\end{tabular}
}
```

Sl.no	Subject	Title	Examination		
			Marks	IA marks	Total
Total					

Notice the braces right before the command `\rowcolors{4}{green!80!yellow!50}{green!70!yellow!40}` and after the tabular environment. The command `\rowcolors` takes three parameters each passed inside braces:

- the row to start,
- the colour for odd rows and
- the colour for even rows.

To colour a single cell, `\cellcolor{blue!10}` command is used in this example.

See the xcolor package documentation (at the further reading section) for a list of available colours and how to create your own. In the example the colours green and yellow are mixed in different proportions.

2.10 Reference guide

quick description of parameters in the tabular environment

Tables can be created using tabular environment.

```
\begin{tabular}[pos]{cols}
table content
\end{tabular}
```

where options can be:

- **pos** : Vertical position. It can assume the following values:

t	the line at the top is aligned with the text baseline
b	the line at the bottom is aligned with the text baseline
c or none	the table is centred to the text baseline

- **cols** : Defines the alignment and the borders of each column. It can have the following values:

l	left-justified column
c	centred 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
*{num}{form}	the format form is repeated num times; for example *{3}{|l|} is equal to |l|l|l|

To separate between cells and introducing new lines use the following commands:

&	column separator
\\\\	start new row (additional space may be specified after \\\\ using square brackets, such as \\\\ [6pt])
`\hline`	horizontal line between rows
`\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

3. Inserting Images

Images are essential elements in most of the scientific documents. LATEX provides several options to handle images and make them look exactly what you need. In this article is explained how to include images in the most common formats, how to shrink, enlarge and rotate them, and how to reference them within your document.

3.1 Introduction

Below is a example on how to import a picture.

```
\documentclass{article}
\usepackage{graphicx}
\graphicspath{ {Pictures/} }

\begin{document}

\includegraphics{pi}

There's a picture of a raspberry pi above
\end{document}
```

Latex can not manage images by itself, so we need to use the graphicx package. To use it, we include the following line in the preamble: `\usepackage{graphicx}`

The command `\graphicspath{ {Pictures/} }` tells LATEX that the images are kept in a folder named images under the current directory.

The `\includegraphics{pi}` command is the one that actually included the image in the document. Here pi is the name of the file containing the image without the extension, then pi.PNG becomes pi. The file name of the image should not contain white spaces nor multiple dots.

3.2 The folder path to images

When working on a document which includes several images it's possible to keep those images in one or more separated folders so that your project is more organised.

In the example at the introduction the command `\graphicspath{ {Pictures/} }` tells LATEX to look in the Pictures folder. The path is relative to the current working directory.

The path to the folder can be relative (recommended) if it is in the same location as the main .tex file or in one of the sub-folders, or absolute if you have to specify the exact path. For example:

```
%Path in Windows format:
\graphicspath{ {c:/user/images/} }

%Path in Unix-like (Linux, OsX) format
\graphicspath{ {/home/user/images/} }
```

Notice that this command requires a trailing slash / and that the path is in between double braces.

You can also set multiple paths if the images are saved in more than one folder. For instance, if there are two folders named images1 and images2, use the command.

```
\graphicspath{ {images1/}{images2/} }
```

If no path is set LATEX will look for pictures in the folder where the .tex file is saved.

3.3 Changing the image size and rotating the picture

If we want to further specify how LATEX should include our image in the document (length, height, etc), we can pass those settings in the following format.

```
\includegraphics[scale=1.5]{pi}
```

The command `\includegraphics[scale=1.5]{pi}` will include the image pi in the document, the extra parameter scale=1.5 will do exactly that, scale the image 1.5 of its real size.

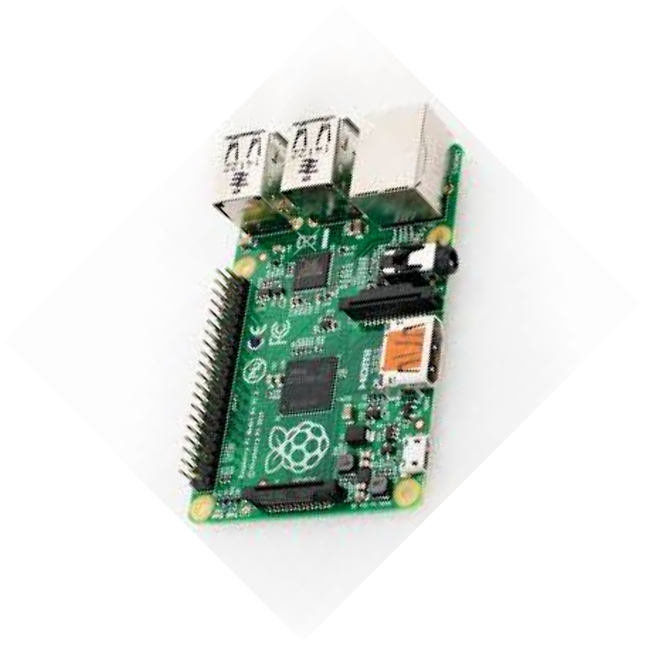
As you probably have guessed, the parameters inside the brackets [width=3cm, height=4cm] define the width and the height of the picture. You can use different units for these parameters. If only the width parameter is passed, the height will be scaled to keep the aspect ratio.

The length units can also be relative to some elements in document. If you want, for instance, make a picture the same width as the text:

```
\includegraphics[width=\textwidth]{pi}
```

There is another common option when including a picture within your document, to rotate it. This can easily accomplished in LATEX:

```
\includegraphics[scale=1, angle=45]{pi}
```



The parameter angle=45 rotates the picture 45 degrees counter-clockwise. To rotate the picture clockwise use a negative number.

3.4 Positioning

In the previous section was explained how to include images in your document, but the combination of text and images may not look as we expected. To change this we need to introduce a new environment.

```
\begin{figure}[h]
\includegraphics[width=8cm]{pi}
\end{figure}
```

The figure environment is used to display pictures as floating elements within the document. This means you include the picture inside the figure environment and you don't have to worry about it's placement, LATEX will position it in a such way that it fits the flow of the document.

Anyway, sometimes we need to have more control on the way the figures are displayed. An additional parameter can be passed to determine the figure positioning. In the example, beginfigure[h], the parameter inside the brackets set the position of the figure to here. Below a table to list the possible positioning values.

Parameter	Position
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!.

3.5 Captioning, labelling and referencing

Captioning images to add a brief description and labelling them for further reference are two important tools when working on a lengthy text.

3.5.1 Captions

Let's start with a caption example:

```
\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{pi}
\caption{Raspberry pi}
\end{figure}
```

Figure 3.1: Raspberry pi

It's really easy, just add the `\caption{Some caption}` and inside the braces write the text to be shown. The placement of the caption depends on where you place the command; if it'a above the includegraphics then the caption will be on top of it, if it's below then the caption will also be set below the figure.

3.5.2 Labels and cross-references

Figures, just as many other elements in a LATEX document (equations, tables, plots, etc) can be referenced within the text. This is very easy, just add a **label** to the *figure* or *SCfigure* environment, then later use that label to refer the picture

```
\begin{figure}[h]
\centering
\includegraphics[width=0.25\textwidth]{xbee}
\caption{Xbee}
\label{fig:xbee1}
\end{figure}

As you can see in the figure \ref{fig:xbee1}...... Also, in the page \pageref{fig:xbee1}
is the same example.
```

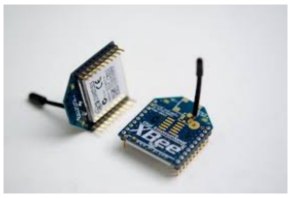


Figure 3.2: Xbee

There are three commands that generate cross-references in this example.

`\label{fig:xbee1}`
This will set a label for this figure. Since labels can be used in several types of elements within the document, it's a good practice to use a prefix, such as *fig:* in the example.

`\ref{fig:xbee1}`
This command will insert the number assigned to the figure. It's automatically generated and will be updated if insert some other figure before the referenced one.

`\pageref{fig:xbee1}`
This prints out the page number where the referenced image appears.

The `\caption` is mandatory to reference a figure.

Another great characteristic in a LATEX document is the ability to automatically generate a list of figures. This is straightforward.

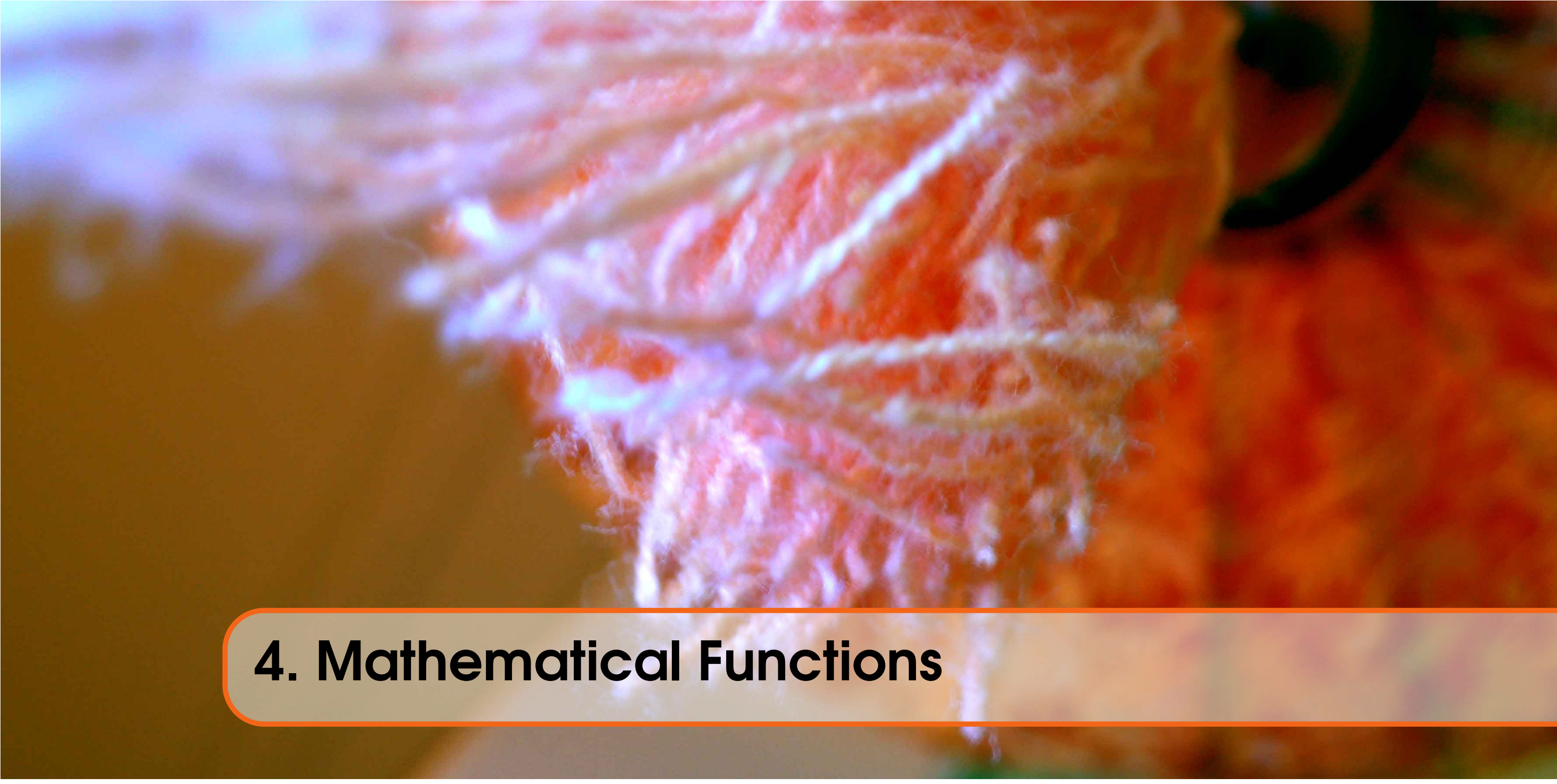
`\listoffigures`

This command only works on captioned figures, since it uses the caption in the table. The example above lists the images in this article.

3.6 Reference guide

LATEX units and legths

Abbreviation	**Definition**
pt	A point, is the default length unit. About 0.3515mm
mm	a millimetre
cm	a centimetre
in	an inch
ex	the height of an x in the current font
em	the width of an m in the current font
`\columnsep`	distance between columns
`\columnwidth`	width of the column
`\linewidth`	width of the line in the current environment
`\paperwidth`	width of the page
`\paperheight`	height of the page
`\textwidth`	width of the text
`\textheight`	height of the text
`\unitleght`	units of length in the picture environment.



4. Mathematical Functions

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 1 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.

4.1 Mathematics Environments

LaTeX needs to know beforehand that the subsequent text does indeed contain mathe- matical 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 math requires 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

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.

4.2 Subscripts and superscripts

The use of superscripts and subscripts is very common in mathematical expressions involving exponents, indexes, and in some special operators. This article explains how to write superscripts and subscripts in simple expressions, integrals, summations, etcetera.

Definite integrals are some of the most common mathematical expressions, let's see an example:

```
\[ \int\limits_0^1 x^2 + y^2 \ dx \]
```

$$\int\limits_0^1 x^2 + y^2 \ dx$$

In LaTeX, subscripts and superscripts are written using the symbols ând _, in this case the x and y exponents where written using these codes. The codes can also be used in some types of mathematical symbols, in the integral included in the example the _ is used to set the lower bound and the ˆfor the upper bound. The command `\limits` changes the way the limits are displayed in the integral, if not present the limits would be next to the integral symbol instead of being on top and bottom.

The symbols _ and ĉan also be combined in the same expression, for example:

```
\[ a_1^2 + a_2^2 = a_3^2 \]
```

$$a_1^2 + a_2^2 = a_3^2$$

If the expression contains long superscripts or subscripts, these need to be collected in braces, as LaTeX normally applies the mathematical commands ând _ only to the following character:

```
\[ x^{2 \alpha} - 1 = y_{ij} + y_{ij}  \]
```

$$x^{2\alpha} - 1 = y_{ij} + y_{ij}$$

Subscripts and superscripts can be nested and combined in various ways. When nesting subscripts/superscripts, however, remember that each command must refer to a single element; this can be a single letter or number, as in the examples above, or a more complex mathematical expression collected in braces or brackets. For example:

```
\[ (a^n)^{r+s} = a^{nr+ns}  \]
```

$$(a^n)^{r+s} = a^{nr+ns}$$

4.3 Brackets and Parentheses

Parentheses and brackets are very common in mathematical formulas. You can easily control the size and style of brackets in LaTeX.

```
\[
\left \{
\begin{tabular}{ccc}
1 & 5 & 8 \\
0 & 2 & 4 \\
3 & 3 & -8
\end{tabular}
\right \}
\]
```

$$\left\{ \begin{array}{ccc} 1 & 5 & 8 \\ 0 & 2 & 4 \\ 3 & 3 & \text{-8} \end{array} \right\}$$

Notice that to insert the brackets, the `\left` and `\right` commands are used. Even if you are using only one bracket both commands are mandatory.

LaTeXmarkup	Renders as
`\big( \Big( \bigg( \Bigg(`	$\big(\Big(\bigg(\Bigg($
`\big] \Big] \bigg] \Bigg]`	$\big] \Big] \bigg] \Bigg]$
`\big\{ \Big\{ \bigg\{ \Bigg\{`	$\big\{ \Big\{ \bigg\{ \Bigg\{$
`\big \langle \Big \langle \bigg \langle \Bigg \langle`	$\big \langle \Big \langle \bigg \langle \Bigg \langle$
`\big \rangle \Big \rangle \bigg \rangle \Bigg \rangle`	$\big \rangle \Big \rangle \bigg \rangle \Bigg \rangle$

4.4 Fractions and Binomials

Fractions and binomial coefficients are common mathematical elements with similar characteristics, one number goes on top of other.

Using fractions and binomial coefficients in an expression is straightforward.

```
The binomial coefficient is defined by the next expression:

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

The binomial coefficient is defined by the next expression:

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

For these commands to work you must import the package amsmath by adding the next line to the preamble of your file

```
\usepackage{amsmath}
```

4.4.1 Displaying fractions

The appearance of the fraction may change depending on the context

```
Fractions can be used alongside the text, for
example \( \frac{1}{2} \), and in a mathematical
display style like the one below:

\[\frac{1}{2}\]
```

Fractions can be used alongside the text, for example $\frac{1}{2}$, and in a mathematical display style like the one below:

$$\frac{1}{2}$$

As you may have guessed, the command `\frac{1}{2}` is the one that displays the fraction. The text inside the first pair of braces is the numerator and the text inside the second pair is the denominator.

Also, the text size of the fraction changes according to the text around it. You can set this manually if you want.

```
When displaying fractions in-line, for example \(\frac{3x}{2}\)
you can set a different display style:
\( \displaystyle \frac{3x}{2} \).

This is also true the other way around

\[ f(x)=\frac{P(x)}{Q(x)} \ \ \textrm{and}
\ \ f(x)=\textstyle\frac{P(x)}{Q(x)} \]
```

When displaying fractions in-line, for example $\frac{3x}{2}$ you can set a different display style: $\displaystyle \frac{3x}{2}$.

This is also true the other way around

$$f(x)=\frac{P(x)}{Q(x)} \ \ \textrm{and} \ \ f(x)=\textstyle\frac{P(x)}{Q(x)}$$

The command `\displaystyle` will format the fraction as if it were in mathematical display mode. On the other side, `\textstyle` will change the style of the fraction as if it were part of the text.

4.4.2 Continued fractions

The usage of fractions is quite flexible, they can be nested to obtain more complex expressions.

```
The fractions can be nested

\[ \frac{1+\frac{a}{b}}{1+\frac{1}{1+\frac{1}{a}}} \]

Now a wild example

\[
a_0+\cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cdots}}}
\]
```

The fractions can be nested

$$\frac{1+\frac{a}{b}}{1+\frac{1}{1+\frac{1}{a}}}$$

Now a wild example

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$$

The second fraction displayed in the previous example uses the command `\cfrac{}{}` provided by the package amsmath, this command displays nested fractions without changing the size of the font. Specially useful for continued fractions.

4.4.3 Binomial coefficients

Binomial coefficients are common elements in mathematical expressions, the command to display them in LaTeX is very similar to the one used for fractions.

```
The binomial coefficient is defined by the next expression:

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

And of course this command can be included in the normal
text flow \(\binom{n}{k}\).
```

The binomial coefficient is defined by the next expression:

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

And of course this command can be included in the normal text flow $\binom{n}{k}$.

As you see, the command `\binom{}{}` will print the binomial coeficient using the parameters passed inside the braces.

4.5 Aligning equations with amsmath

The amsmath package provides a handful of options for displaying equations. You can choose the layout that better suits your document, even if the equations are really long, or if you have to include several equations in the same line.

The standard LaTeX tools for equations may lack some flexibility, causing overlapping or even trimming part of the equation when it's too long. We can surpass this difficulties with amsmath. Let's check an example:

```
\begin{equation} \label{eq1}
\begin{split}
A & = \frac{\pi r^2}{2} \\
& = \frac{1}{2} \pi r^2
\end{split}
\end{equation}
```

$$\begin{split} A &= \frac{\pi r^2}{2} \\ &= \frac{1}{2}\pi r^2 \end{split} \tag{4.1}$$

You have to wrap you equation in the equation environment if you want it to be numbered, use equation* (with an asterisk) otherwise. Inside the equation environment use the split environment to split the equations into smaller pieces, these smaller pieces will be aligned accordingly. The double backslash works as a newline character. Use the ampersand character &, to set the points where the equations are vertically aligned.

4.5.1 Writing a single equation

To display a single equation, as mentioned in the introduction, you have to use the equation* or equation environment, depending on whether you want the equation to be numbered or not. Additionally you might add a label for future reference within the document.

```
\begin{equation} \label{eu_eqn}
e^{\pi i} + 1 = 0
\end{equation}

The beautiful equation \ref{eu_eqn} is known as the Euler equation
```

$$e^{\pi i} + 1 = 0 \tag{4.2}$$

The beautiful equation 4.2 is known as the Euler equation

4.5.2 Displaying long equations

For equations longer than a line use the *multline* environment. Insert a double backslash to set a point for the equation to be broken. The first part will be aligned to the left and the second part will be displayed in the next line and aligned to the right.

Again, the use of an asterisk * in the environment name determines whether the equation is numbered or not.

```
\begin{multline*}
p(x) = 3x^6 + 14x^5y + 590x^4y^2 + 19x^3y^3\\
- 12x^2y^4 - 12xy^5 + 2y^6 - a^3b^3
\end{multline*}
```

$$\begin{multline*} p(x) = 3x^6 + 14x^5y + 590x^4y^2 + 19x^3y^3 \\ - 12x^2y^4 - 12xy^5 + 2y^6 - a^3b^3 \end{multline*}$$

4.5.3 Splitting and aligning an equation

Split is very similar to multline. Use the split environment to break an equation and to align it in columns, just as if the parts of the equation were in a table. This environment must be used inside an equation environment. For an example check the introduction of this document.

Aligning several equations

If there are several equations that you need to align vertically, the align environment will do it:

```
\begin{align*}
2x - 5y &=  8 \\
3x + 9y &=  -12
\end{align*}
```

$$\begin{aligned} 2x-5y &= 8 \\ 3x+9y &= -12 \end{aligned}$$

Usually the binary operators (>, < and =) are the ones aligned for a nice-looking document.

As mentioned before, the ampersand character & determines where the equations align. Let's check a more complex example:

```
\begin{align*}
x&=y           &  w &=z              &  a&=b+c\\
2x&=-y         &  3w&=\frac{1}{2}z   &  a&=b\\
-4 + 5x&=2+y   &  w+2&=-1+w          &  ab&=cb
\end{align*}
```

$$\begin{aligned} x&=y & w&=z & a&=b+c\\ 2x&=-y & 3w&=\frac{1}{2}z & a&=b\\ -4+5x&=2+y & w+2&=-1+w & ab&=cb \end{aligned}$$

Here we arrange the equations in three columns. LaTeX assumes that each equation consists of two parts separated by a &; also that each equation is separated from the one before by an &.

Again, use * to toggle the equation numbering. When numbering is allowed, you can label each row individually.

Grouping and centering equations

If you just need to display a set of consecutive equations, centered and with no alignment whatsoever, use the gather environment. The asterisk trick to set/unset the numbering of equations also works here.

```
\begin{gather*}
2x - 5y =  8 \\
3x^2 + 9y =  3a + c
\end{gather*}
```

$$\begin{gathered} 2x-5y=8 \\ 3x^2+9y=3a+c \end{gathered}$$

4.6 Operators

Characters in mathematical mode are usually shown in italics, but sometimes especial function names require different formatting, this is accomplished by using operators defined in LaTeX.

Trigonometrical functions, logarithms, and some others can be written in a document by means of some special commands.

```
Examples of mathematical operators

\[
\sin(a + b ) = \sin(a)\cos(b) + \cos(a)\sin(a)
\]
```

Examples of mathematical operators

$$\sin(a+b) = \sin(a)\cos(b)+\cos(a)\sin(a)$$

The commands will print the name of the function in Roman characters instead of italics.

Some operators can take parameters that are handled in a special way, for instance, limits.

```
Testing notation for limits

\[
\lim_{h \rightarrow 0 } \frac{f(x+h)-f(x)}{h}
\]

This operator changes when used alongside
text \( \lim_{x \rightarrow h} (x-h) \).
```

Testing notation for limits

$$\lim_{h \rightarrow 0} \frac{f(x+h)-f(x)}{h}$$

This operator changes when used alongside text $\lim_{x \rightarrow h}(x-h)$.

Notice how the limit declaration can include a subscript. See the reference guide for a complete list of available operators.

4.7 Spacing in math mode

In mathematical mode characters are spaced as if they were part of a single word, regardless of the actual space you insert. This article explains how to insert spaces of different lengths in mathematical mode.

Spacing in maths mode is useful in several situations, let's see an example:

```
Assume we have the next sets
\[
S = \{ z \in \mathbb{C}\, |\, |z| < 1 \} \quad \textrm{and} \quad S_2=\partial{S}
\]
```

Assume we have the next sets

$$S = \{z \in \mathbb{C} \,|\, |z| < 1\} \quad \text{and} \quad S_2 = \partial S$$

As you see in this example, a mathematical text can be explicitly spaced by means of some special commands.

4.7.1 Spaces

The spacing depends on the command you insert, the example below contains a complete list of spaces and how they look like.

```
Spaces in mathematical mode.

\begin{align*}
f(x) =& x^2\! +3x\! +2 \\
f(x) =& x^2+3x+2 \\
f(x) =& x^2\, +3x\, +2 \\
f(x) =& x^2\: +3x\: +2 \\
f(x) =& x^2\; +3x\; +2 \\
f(x) =& x^2\ +3x\ +2 \\
f(x) =& x^2\quad +3x\quad +2 \\
f(x) =& x^2\qquad +3x\qquad +2
\end{align*}
```

Spaces in mathematical mode.

$$\begin{align*}
f(x) =& x^2\! +3x\! +2 \\
f(x) =& x^2+3x+2 \\
f(x) =& x^2\, +3x\, +2 \\
f(x) =& x^2\: +3x\: +2 \\
f(x) =& x^2\; +3x\; +2 \\
f(x) =& x^2\ +3x\ +2 \\
f(x) =& x^2\quad +3x\quad +2 \\
f(x) =& x^2\qquad +3x\qquad +2
\end{align*}$$

4.8 Integrals, sums and limits

4.8.1 Integrals

Integral expression can be added using the command

```
\int_{lower}^{upper}
```

Note, that integral expression may seems a little different in *inline* and *display* math mode - in inline mode the integral symbol and the limits are compressed.

LaTeX code	Output
Integral `$\int_{a}^{b} x^2 dx$` inside text	Integral $\int_a^b x^2 dx$ inside text
`$$\int_{a}^{b} x^2 dx$$`	$$\int_{a}^{b} x^2 dx$$

4.8.2 Multiple integrals

To obtain double/triple/multiple integrals and cyclic integrals you must use amsmath and esint (for cyclic integrals) packages.

LaTeX code	Output
`$$\iint_V \mu(u,v) \,du\,dv$$` `$$\iiint_V \mu(u,v,w) \,du\,dv\,dw$$` `$$\iiiint_V \mu(t,u,v,w) \,dt\,du\,dv\,dw$$` `$$\idotsint_V \mu(u_1,\dots,u_k) \,du_1 \dots du_k$$`	$$\iint_V \mu(u,v) \,du\,dv$$ $$\iiint_V \mu(u,v,w) \,du\,dv\,dw$$ $$\iiiint_V \mu(t,u,v,w) \,dt\,du\,dv\,dw$$ $$\idotsint_V \mu(u_1,\dots,u_k) \,du_1 \dots du_k$$
`$$\oint_V f(s) \,ds$$` `$$\oiint_V f(s,t) \,ds\,dt$$`	$$\oint_V f(s) \,ds$$ $$\oiint_V f(s,t) \,ds\,dt$$

4.8.3 Sums and products

Like integral, sum expression can be added using the `\sum_{lower}^{upper}` command.

LaTeX code	Output
`Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text`	Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text
`$$\sum_{n=1}^{\infty} 2^{-n} = 1$$`	$$\sum_{n=1}^{\infty} 2^{-n} = 1$$

In similar way you can obtain expression with product of a sequence of factors using the `\prod_{lower}^{upper}` command.

LaTeX code	Output
`Product $\prod_{i=a}^{b} f(i)$ inside text`	Product $\prod_{i=a}^{b} f(i)$ inside text
`$$\prod_{i=a}^{b} f(i)$$`	$$\prod_{i=a}^{b} f(i)$$

4.8.4 Limits

Limit expression can be added using the `\lim_{lower}` command.

LaTeX code	Output
`Limit $\lim_{x\to\infty} f(x)$ inside text`	Limit $\lim_{x\to\infty} f(x)$ inside text
`$$\lim_{x\to\infty} f(x)$$`	$$\lim_{x\to\infty} f(x)$$

4.8.5 Integer and sum limits improvement

In inline math mode the integral/sum/product lower and upper limits are placed right of integral symbol. Similar is for limit expressions. If you want the limits of an integral/sum/product to be specified above and below the symbol in inline math mode, use the `\limits` command before limits specification.

$\LaTeX$code	Output
`Integral $\int_{a}^{b} x^2 dx$ inside text`	Integral $\int_{a}^{b} x^2 dx$ inside text
`Improved integral $\int\limits_{a}^{b} x^2 dx$ inside text`	Improved integral $\int\limits_{a}^{b} x^2 dx$ inside text
`Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text`	Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text
`Improved sum $\sum\limits_{n=1}^{\infty} 2^{-n} = 1$ inside text`	Improved sum $\sum\limits_{n=1}^{\infty} 2^{-n} = 1$ inside text

4.9 List of Greek letters and math symbols

Greek letters

αA	`\alpha A`	νN	`\nu N`
βB	`\beta B`	$\xi\Xi$	`\xi\Xi`
$\gamma\Gamma$	`\gamma \Gamma`	$o O$	`o O`
$\delta\Delta$	`\delta \Delta`	$\pi\Pi$	`\pi \Pi`
$\epsilon\varepsilon E$	`\epsilon \varepsilon E`	$\rho\varrho P$	`\rho\varrho P`
ζZ	`\zeta Z`	$\sigma\Sigma$	`\sigma \Sigma`
ηH	`\eta H`	τT	`\tau T`
$\theta\vartheta\Theta$	`\theta \vartheta \Theta`	$\upsilon\Upsilon$	`\upsilon \Upsilon`
ιI	`\iota I`	$\phi\varphi\Phi$	`\phi \varphi \Phi`
κK	`\kappa K`	χX	`\chi X`
$\lambda\Lambda$	`\lambda \Lambda`	$\psi\Psi$	`\psi \Psi`
μM	`\mu M`	$\omega\Omega$	`\omega \Omega`

Arrows

$\leftarrow$	`\leftarrow`	$\Leftarrow$	`\Leftarrow`
$\rightarrow$	`\rightarrow`	$\Rightarrow$	`\Rightarrow`
$\leftrightarrow$	`\leftrightarrow`	$\rightleftharpoons$	`\rightleftharpoons`
$\uparrow$	`\uparrow`	$\downarrow$	`\downarrow`
$\Uparrow$	`\Uparrow`	$\Downarrow$	`\Downarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`
$\nearrow$	`\nearrow`	$\searrow$	`\searrow`
$\swarrow$	`\swarrow`	$\nwarrow$	`\nwarrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`
$\rightleftharpoons$	`\rightleftharpoons`		

5. Flowcharts

5.1 Introduction

In this chapter we're going to be looking at creating flowcharts in TikZ. To get started we need to load up the TikZ package, the 'shapes.geometric' TikZ library and the 'arrows' library.

```
\usepackage{tikz}
\usetikzlibrary{shapes.geometric, arrows}
```

5.2 Defining Blocks

Before we start the document we need to define the basic components of a flow chart. To do this we use the `\tikzstyle` command. First let's define the block we're going to use for start and stop blocks. We'll name it 'startstop' using curly brackets immediately following the command, then we add an equals sign before a set of square brackets. In the square brackets we enter all the formatting information. For this block we'll specify a rectangle with rounded corners. We'll give it a minimum width of 3cm and a minimum height of 1cm. We'll also ensure the text gets centred and we'll set both a draw and a fill colour. In this example we've set the fill colour to a colour that is 30% red mixed with 70% white.

```
\tikzstyle{startstop} = [rectangle,rounded corners,minimum width=3cm,
minimum height=1cm,text centered,draw=black,fill=red!30]
```

Next we'll specify an input or output box. This time we want the block to be a parallelogram. To achieve this we ask for a trapezium and then alter the angles. The rest is very similar.

```
\tikzstyle{io} = [trapezium, trapezium left angle=70,
 trapezium right angle=110, minimum width=3cm, minimum height=1cm,
 text centered, draw=black, fill=blue!30]
```

Next we'll add a TikZ style for process blocks using a rectangle and a style for decision blocks using a diamond.

```
\tikzstyle{process} = [rectangle, minimum width=3cm, minimum height=1cm,
text centered, draw=black, fill=orange!30]

\tikzstyle{decision} = [diamond, minimum width=3cm, minimum height=1cm,
text centered, draw=black, fill=green!30]
```

5.3 Defining Arrows

Finally we'll define a style for the arrows. For this we set the line thickness to 'thick', add an arrow head and specify the stealth arrow head.

```
\tikzstyle{arrow} = [thick,->,>=stealth]
```

5.4 Building a Flowchart

Now we are ready to start building our flow chart. To do the we use the 'tikzpicture' environment. We'll create our flowchart blocks using nodes and the tikzstyles we defined earlier. Nodes are very powerful as we can easily position them, make them draw a shape, heavily format them and give them some text. In square brackets at the end of the begin command we specify a node distance of 2cm. This is so that the nodes we use to build the blocks are automatically spaced 2cm apart from their centres.

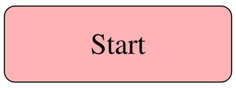
```
\begin{tikzpicture}[node distance=2cm]

<TikZ code>

\end{tikzpicture}
```

5.5 Adding Nodes

To add a node we use the `\node` command. We then add a label for the node in parenthesis. This label is how we refer to the node in the rest of the code. Then in square brackets we add the name of the tikzstyle we want the node to conform to, along with any other formatting options. Then in curly brackets we add the text we want to appear in the block before closing the statement with a semicolon.

```
\node (start) [startstop] {Start};
```

If we now compile the code you'll see our start block has appeared as expected.



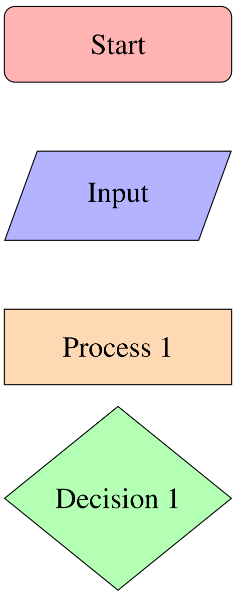
Now let's add an input block in below the start block. This time we need to tell the node where to position itself. To do this we enter 'below of' followed by an equals sign and a node label into the square brackets. We could also use 'above of', 'right of' or 'left of' if we wanted the block to appear somewhere else. We'll tell it to position itself below the start block.

```
\node (in1) [io, below of=start] {Input};
```

Now lets add in a process block and a decision block.

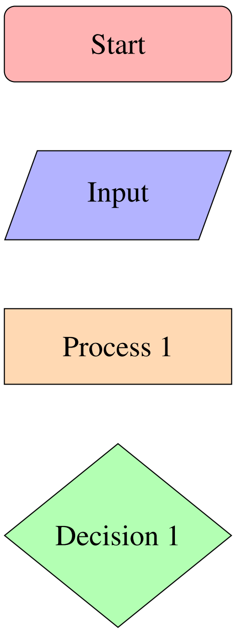
```
\node (pro1) [process, below of=in1] {Process 1};
\node (dec1) [decision, below of=pro1] {Decision 1};
```

If we compile the code you'll notice that the gap between the green decision block and the orange process block isn't as big as the other gaps.



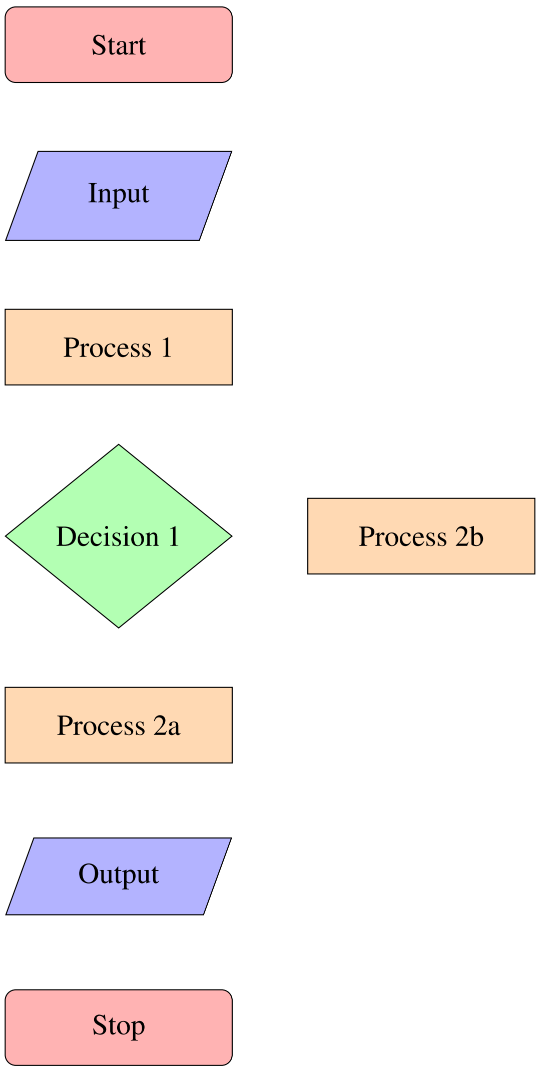
This is because the decision block, being a diamond, is taller than the other blocks. Therefore we can manually adjust its position using the 'yshift' variable. If we enter `yshift=-0.5cm` it will move the decision block vertically down by 0.5cm which should make the gap more regular.

```
\node (dec1) [decision, below of=pro1, yshift=-0.5cm] {Decision 1};
```



Now lets add in two process blocks coming out of the decision block, one below it and one to the right of it. Again we'll need to alter the positioning using 'yshift' for the block below and 'xshift' for the block to the right. Let's finish off adding the blocks by adding in an output block and a stop block.

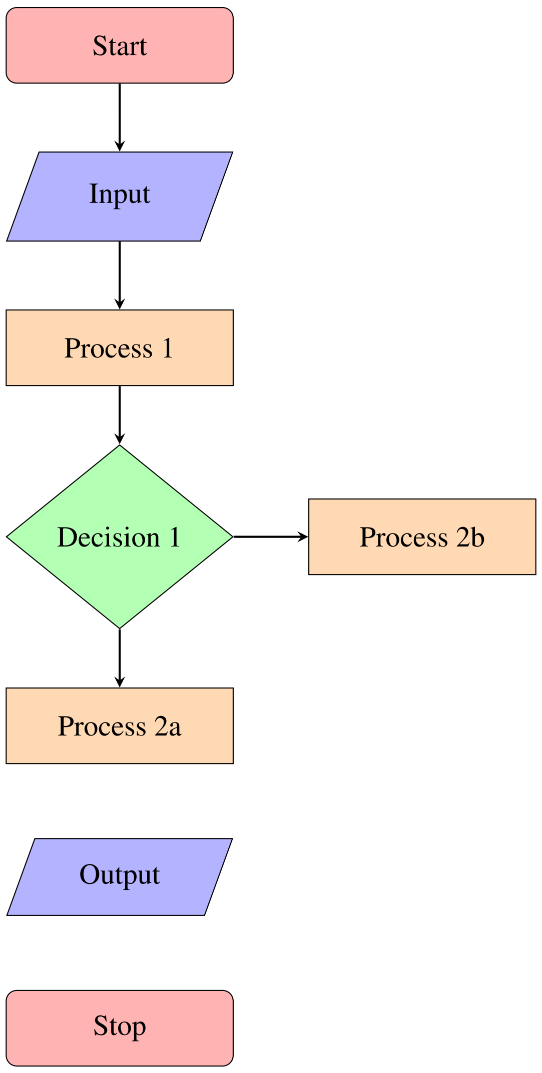
```
\node (pro2a) [process, below of=dec1, yshift=-0.5cm] {Process 2a};
\node (pro2b) [process, right of=dec1, xshift=2cm] {Process 2b};
\node (out1) [io, below of=pro2a] {Output};
\node (stop) [startstop, below of=out1] {Stop};
```



5.6 Adding Arrows

To finish off our flowchart we need to add the arrows in. To draw an arrow we use the `\draw` command and then specify the tikzstyle we prepared for arrows using square brackets. We then enter the label of the node we want the arrow to start from, followed by two dashes and then the label corresponding to the node we want the arrow to terminate at. The labels need to be in parenthesis and we need to make sure we close the statement with a semicolon. Lets add arrows in between the start block and the input block, the input and process 1, process 1 and decision 1, decision 1 and process 2a and between decision 1 and process 2b.

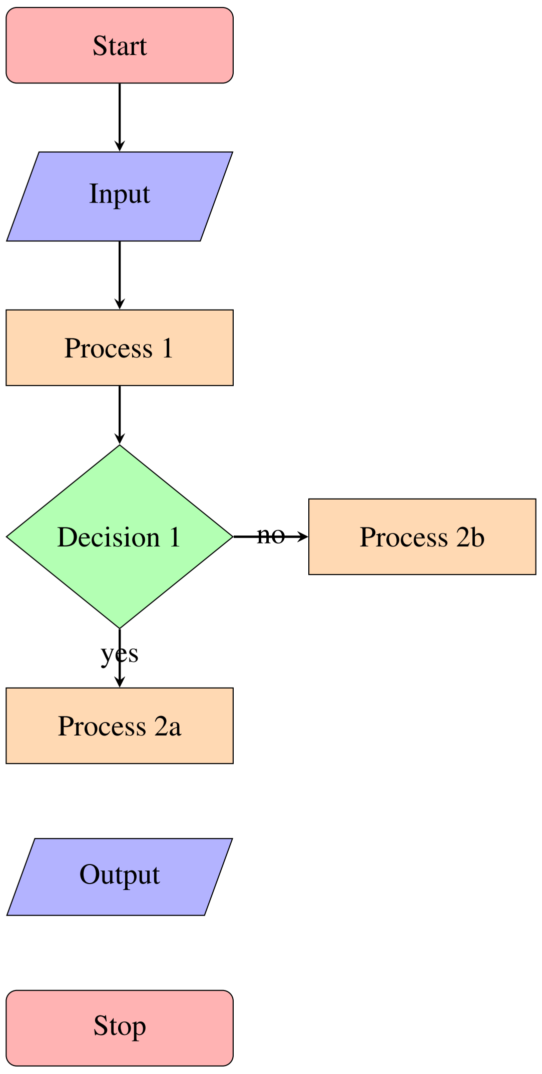
```
 \draw [arrow] (start) -- (in1);
 \draw [arrow] (in1) -- (pro1);
 \draw [arrow] (pro1) -- (dec1);
 \draw [arrow] (dec1) -- (pro2a);
 \draw [arrow] (dec1) -- (pro2b);
```



As we have arrows coming out of a decision block we need to add some text to these two arrows. To do this we use more nodes, however this time we don't need to use the `\node` command, we just type the word node in after the two dashes and then the text in curly brackets.

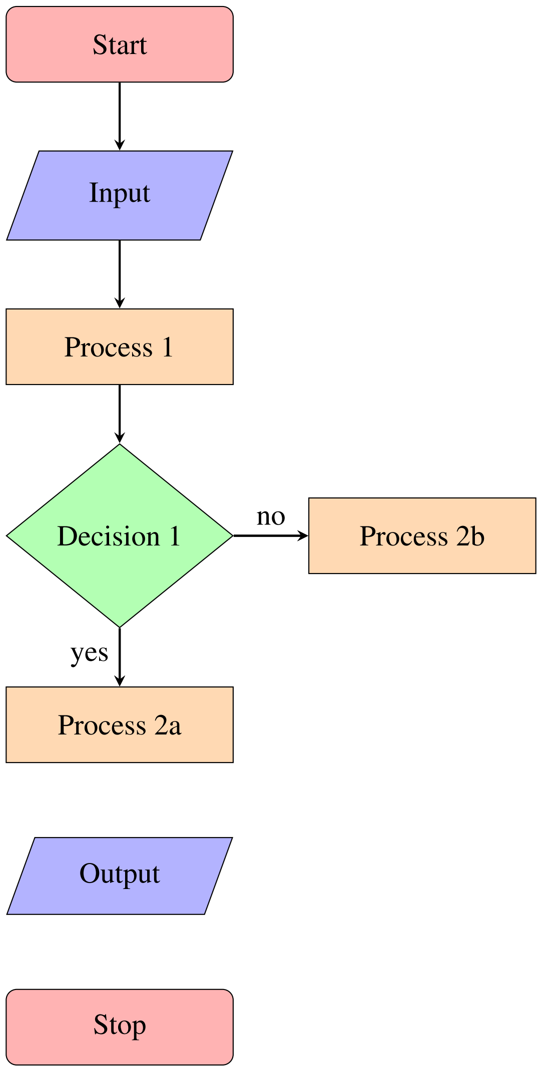
```
\draw [arrow] (dec1) -- node {yes} (pro2a);
\draw [arrow] (dec1) -- node {no} (pro2b);
```

If we now compile the code you'll see the text has been added but not in a very helpful place.



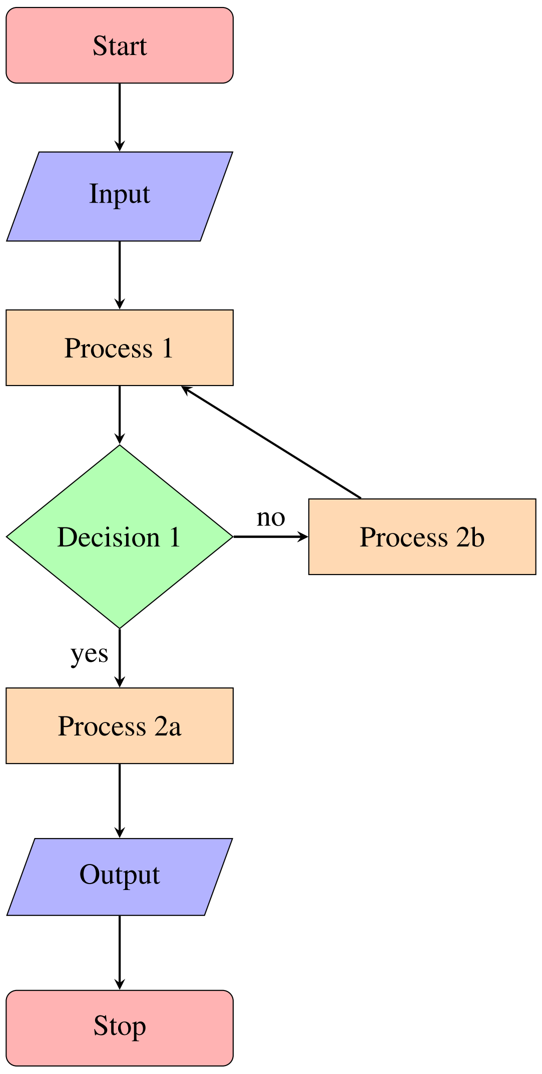
To fix this we specify which of the node's anchors TikZ should use to fix the nodes to the lines. To do this we use square brackets immediately after the keyword 'node' and then enter 'anchor=' followed by the anchor. For the 'yes' node we'll use the east anchor and for the 'no' node we'll use the south anchor.

```
\draw [arrow] (dec1) -- node[anchor=east] {yes} (pro2a);
\draw [arrow] (dec1) -- node[anchor=south] {no} (pro2b);
```



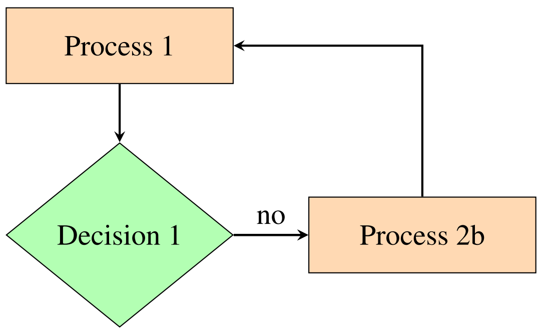
Now let's draw the final arrows in.

```
\draw [arrow] (pro2b) -- (pro1);
\draw [arrow] (pro2a) -- (out1);
\draw [arrow] (out1) -- (stop);
```



You'll also notice that the arrow from process 2b to process 1 is diagonal and therefore doesn't look right. To improve this we can swap the first dash for a bar symbol which will make the arrow go in a vertical direction before going in a horizontal direction.

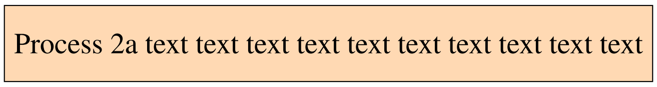
```
\draw [arrow] (pro2b) |- (pro1);
```



5.7 Text Width

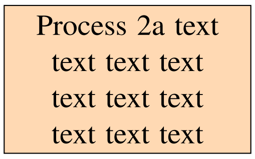
The final thing we should discuss is the text width. At the moment all our text fits nicely inside our shapes. However, if for example, we add some more text to process 2a, you'll see the shape just extends horizontally until the text fits.

```
\node (pro2a) [process, below of=dec1, yshift=-0.5cm]
{Process 2a text text text text text text text text text text};
```



This now becomes a bit messy. To improve it we can specify the text width for these nodes by entering text width= followed by a length into our tikzstyles.

```
\tikzstyle{process} = [rectangle, minimum width=3cm,minimum height=1cm,
 text centered, text width=3cm, draw=black, fill=orange!30]
```



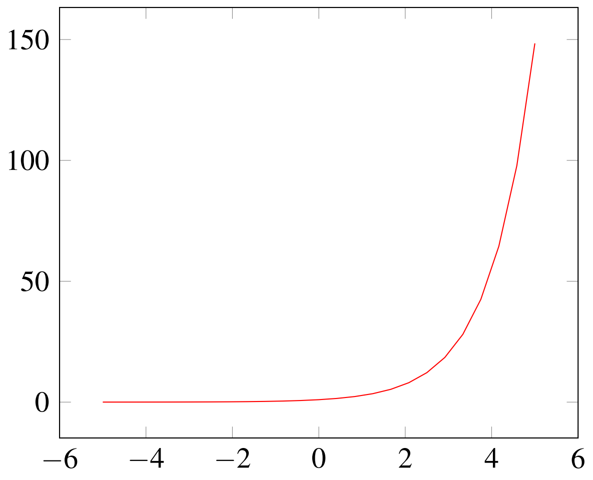
6. Graphs

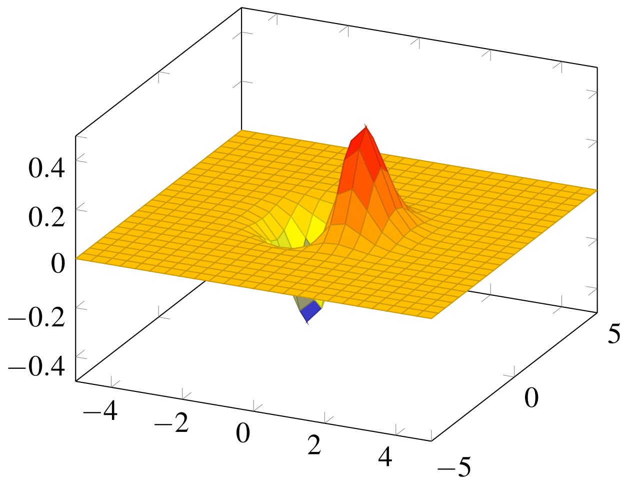
The *pgfplots* package is a powerful tool, based on tikz, dedicated to create scientific graphs.

6.1 Introduction

Pgfplots is a visualization tool to make simpler the inclusion of plots in your documents. The basic idea is that you provide the input data/formula and pgfplots does the rest.

```
\begin{tikzpicture}
\begin{axis}
\addplot[color=red]{exp(x)};
\end{axis}
\end{tikzpicture}
%Here ends the first plot
\hskip 5pt
%Here begins the 3d plot
\begin{tikzpicture}
\begin{axis}
\addplot3[
surf,
]
{exp(-x^2-y^2)*x};
\end{axis}
\end{tikzpicture}
```





Since pgfplot is based on tikz the plot must be inside a tikzpicture environment. Then the environment declaration `\begin{axis}`,`\end{axis}` will set the right scaling for the plot, check the Reference guide for other axis environments.

To add an actual plot, the command `\addplot[color=red]{log(x)};` is used. Inside the squared brackets some options can be passed, in this case we set the colour of the plot to red; the squared brackets are mandatory, if no options are passed leave a blank space between them. Inside the curly brackets you put the function to plot. Is important to remember that this command must end with a semicolon ;.

To put a second plot next to the first one declare a new tikzpicture environment. Do not insert a new line, but a small blank gap, in this case hskip 10pt will insert a 10pt-wide blank space.

The rest of the syntax is the same, except for the

```
\addplot3 [surf,]{exp(-x^2-y^2)*x};
```

This will add a 3dplot, and the option surf inside squared brackets declares that it's a surface plot. The function to plot must be placed inside curly brackets. Again, don't forget to put a semicolon ; at the end of the command.

Note: It's recommended as a good practice to indent the code - see the second plot in the example above - and to add a comma , at the end of each option passed to`\addplot` . This way the code is more readable and is easier to add further options if needed.

6.2 The Document Preamble

To include pgfplots in your document is very easy, add the next line to your preamble and that's it:

```
\usepackage{pgfplots}
```

Some additional tweaking for this package can be made in the preamble. To change the size of each plot and also guarantee compatibility backwards (recommended) add the next line:

```
\pgfplotsset{width=10cm,compat=1.9}
```

This changes the size of each pgfplot figure to 10 cementers, which is huge; you may use different units (pt, mm, in). The compat parameter is for the code to work on the package version 1.9 or latter.

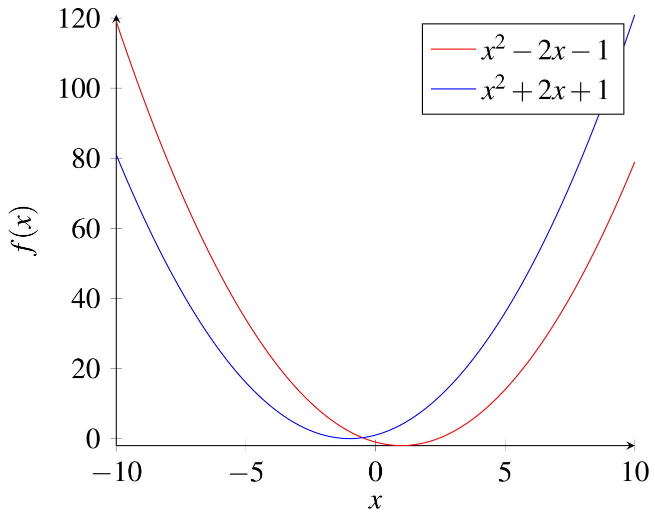
Since LaTeX was not initially conceived with plotting capabilities in mind, when there are several pgfplot figures in your document or they are very complex, it takes a considerable amount of time to render them. To improve the compiling time you can configure the package to export the figures to separate PDF files and then import them into the document, just add the code shown below to the preamble:

```
\usepgfplotslibrary{external}
\tikzexternalize
```

6.3 Plotting Mathematical Expressions

To plot mathematical expressions is really easy:

```
\begin{tikzpicture}
\begin{axis}[
axis lines = left,
xlabel = $x$,
ylabel = {$f(x)$},
]
%Below the red parabola is defined
\addplot [
domain=-10:10,
samples=100,
color=red,
]
{x^2 - 2*x - 1};
\addlegendentry{$x^2 - 2x - 1$}
%Here the blue parabloa is defined
\addplot [
domain=-10:10,
samples=100,
color=blue,
]
{x^2 + 2*x + 1};
\addlegendentry{$x^2 + 2x + 1$}

\end{axis}
\end{tikzpicture}
```



Let's analyse the new commands line by line:

```
axis lines = left
```

This will set the axis only on the left and bottom sides of the plot, instead of the default box. Further customisation options at the reference guide.

```
xlabel = $x$ and ylabel = {$f(x)$}
```

Self-explanatory parameter names, these will let you put a label on the horizontal and vertical axis. Notice the ylabel value in between curly brackets, this brackets tell pgfplots how to group the text. The xlabel could have had brackets too. This is useful for complicated labels that may confuse pgfplot.

```
\addplot.
```

This will add a plot to the axis, general usage was described at the introduction. There are two new parameters in this example.

```
domain=-10:10
```

This establishes the range of values of x.
samples=100.
Determines the number of points in the interval defined by domain. The greater the value of samples the sharper the graph you get, but it will take longer to render.

```
\addlegendentry{$x^2 - 2x - 1$}
```

This adds the legend to identify the function

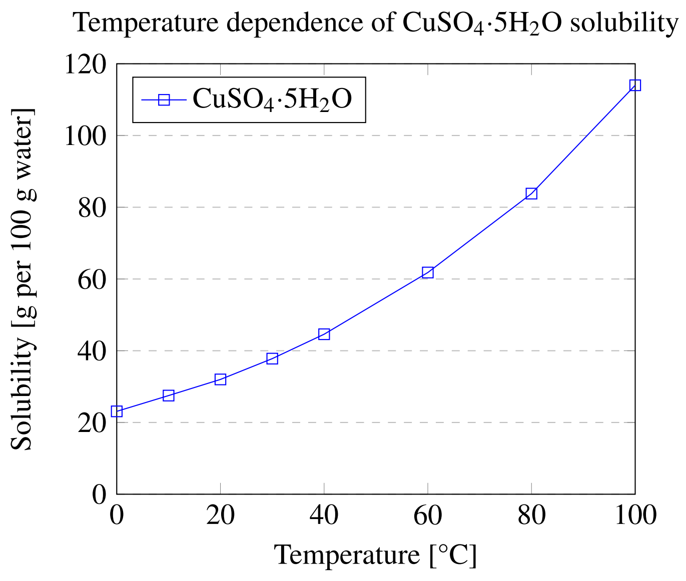
```
x^2 - 2x - 1
```

To add another graph to the plot just write a new `\addplot` entry.

6.4 Plotting from data

Scientific research often yields data that has to be analysed. The next example shows how to plot data with pgfplots:

```
\begin{tikzpicture}
\begin{axis}[
title={Temperature dependence of CuSO$_4\cdot$5H$_2$O solubility},
xlabel={Temperature [\textcelsius]},
ylabel={Solubility [g per 100 g water]},
xmin=0, xmax=100,
ymin=0, ymax=120,
xtick={0,20,40,60,80,100},
ytick={0,20,40,60,80,100,120},
legend pos=north west,
ymajorgrids=true,
grid style=dashed,
]

\addplot[
color=blue,
mark=square,
]
coordinates {
(0,23.1)(10,27.5)(20,32)(30,37.8)(40,44.6)(60,61.8)(80,83.8)(100,114)
};
\legend{CuSO$_4\cdot$5H$_2$O}

\end{axis}
\end{tikzpicture}
```



There are some new commands and parameters here:

```
title={Temperature dependence of CuSO$_4\cdot$5H$_2$O solubility}.
```

As you might expect, assigns a title to the figure. The title will be displayed above the plot.

```
xmin=0, xmax=100, ymin=0, ymax=120
```

Minimum and maximum bounds of the x and y axes.

```
xtick={0,20,40,60,80,100}, ytick={0,20,40,60,80,100,120}
```

Points where the marks are placed. If empty the ticks are set automatically. legend pos=north west. Position of the legend box. Check the reference guide for more options. ymajorgrids=true. This Enables/disables grid lines at the tick positions on the y axis. Use xmajorgrids to enable grid lines on the x axis. grid style=dashed. Self-explanatory. To display dashed grid lines. mark=square. This draws a squared mark at each point in the cordinates array. Each mark will be connected with the next one by a straight line.

```
 Coordinates{(0,23.1)(10,27.5)(20,32)...}
```

Coordinates of the points to be plotted. This is the data you want analyse graphically. If the data is in a file, which is the case most of the time; instead of the commands `\addplot` and coordinates you should use

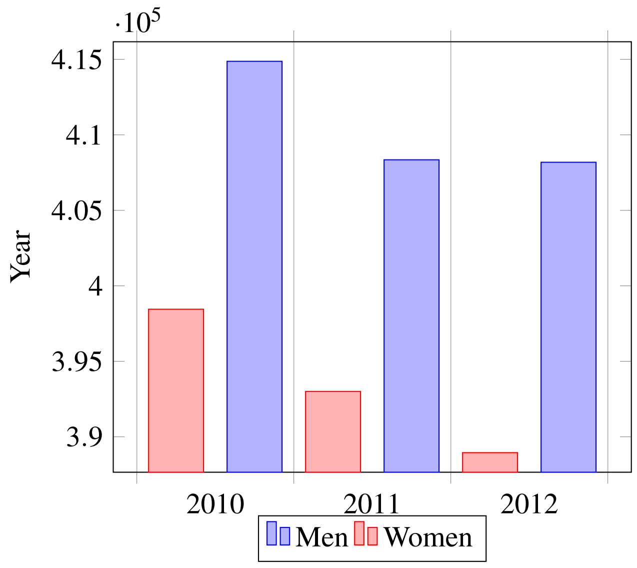
```
 \addplot table {file_with_the_data.dat}
```

the rest of the options are valid in this environment.

6.5 Bar graphs

Bar graphs (also known as bar charts and bar plots) are used to display gathered data, mainly statistical data about a population of some sort. Bar plots in pgfplots are highly customisable, but here we are going to show an example that ‘just works’:

```
\begin{tikzpicture}
\begin{axis}[
x tick label style={
/pgf/number format/1000 sep=},
ylabel=Year,
enlargelimits=0.05,
legend style={at={(0.5,-0.1)},
anchor=north,legend columns=-1},
ybar interval=0.7,
]
\addplot
coordinates {(2012,408184) (2011,408348)
(2010,414870) (2009,412156)};
\addplot
coordinates {(2012,388950) (2011,393007)
(2010,398449) (2009,395972)};
\legend{Men,Women}
\end{axis}
\end{tikzpicture}
```



The figure starts with the already explained declaration of the tikzpicture and axis environments, but the axis declaration has a number of new parameters:

```
x tick label style={/pgf/number format/1000 sep=}
```

This piece of code defines a complete style for the plot. With this style you may include several`\addplot` commands within this axis environment, they will fit and look nice together with no further tweaks (the ybar parameter described below is mandatory for this to work).

```
enlargelimits=0.05.
```

Enlarging the limits in a bar plot is necessary because these kind of plots often require some extra space above the bar to look better and/or add a label. Then number 0.05 is relative to the total height of of the plot.

```
legend style={at={(0.5,-0.2)}, anchor=north,legend columns=-1}
```

Again, this will work just fine most of the time. If anything, change the value of -0.2 to locate the legend closer/farther from the x-axis. ybar interval=0.7, Thickness of each bar. 1 meaning the bars will be one next to the other with no gaps and 0 meaning there will be no bars, but only vertical lines. The coordinates in this kind of plot determine the base point of the bar and its height.

The labels on the y-axis will show up to 4 digits. If in the numbers you are working with are greater than 9999 pgfplot will use the same notation as in the example.

6.6 Pulse Amplitude Modulation - example

- The figure has to be comanded in the following field

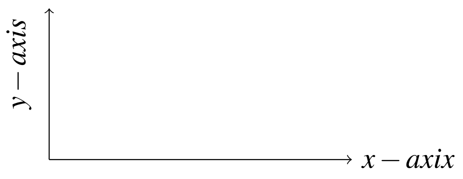
  ```
  \begin{figure}
  \begin{tikzpicture}

  \end{tikzpicture}
  \end{figure}
  ```

- The figure can be scaled up or down with the scale attribute within square brackets.

  ```
  \begin{tikzpicture} [scale=0.7]
  ```

6.6.1 Axes

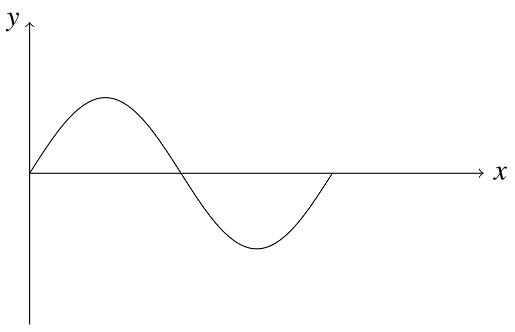
- To draw axis of the graph with naming.



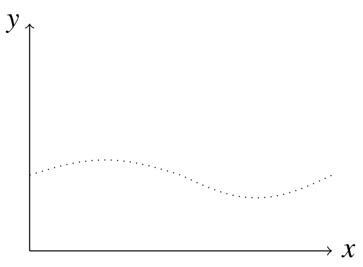
```
\begin{figure}[h!]
\begin{tikzpicture}
\draw[->] (0,0) to (4,0) node[right]{$x-axix$};
\draw[->] (0,0) to(0,2) node[rotate=90,yshift=0.4cm,left]{$y-axis$};
\end{tikzpicture}
\end{figure}
```

- To draw sinusoidal curve.

```
\begin{figure}[h!]
\begin{tikzpicture}
\draw[->] (0,0) to (6,0)node[right]{$x$};
\draw[->] (0,-2) to (0,2)node[left]{$y$};
\draw (0,0) sin (1,1) cos(2,0)sin(3,-1)cos(4,0);
\end{tikzpicture}

\end{figure}
```



- To draw dotted curve.

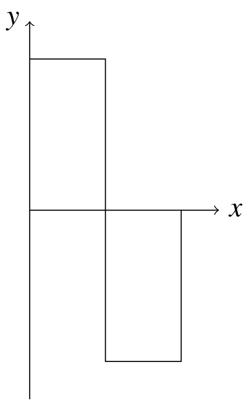


```
\begin{figure}[h!]
\begin{tikzpicture}
```

```
\draw[->] (0,0) to (4,0) node[right]{$x$};
\draw[->] (0,0) to(0,3) node[left]{$y$};
\draw[dotted](0,1)sin(1,1.2)cos(2,1)sin(3,0.7)cos(4,1);
\end{tikzpicture}

\end{figure}
```

- To draw square wave.

```
\begin{figure}[h!]
\begin{tikzpicture}
\draw[->] (0,0) to (2.5,0) node[right]{$x$};
\draw[->] (0,-2.5) to(0,2.5) node[left]{$y$};
\draw (0,2)--(1,2)--(1,0)--(1,-2)--(2,-2)--(2,0);
\end{tikzpicture}

\end{figure}
```



- **sample graph**

```
\begin{figure}
\begin{tikzpicture} [scale=0.7]
\draw[->] (0,-4)--(0,2);
\draw (0,0) node[left]{$r(t)$};
\draw[->] (0,2)--(0,6);
\draw(0,6) node[left]{$PAM$};
\draw(0,14) node[left]{$m(t)$};
\draw(0,12) node[left]{$f_m$};
\draw[->] (0,6)--(0,10);
\draw[->] (0,10)--(0,16);
\draw[->] (0,-4)--(11,-4);
\draw (11,-4) node[right]{$t$};
\draw (11,2) node[right]{$t$};
\draw (11,8) node[right]{$t$};
\draw (11,12) node[right]{$t$};
\draw[->] (0,2)--(11,2);
\draw[->] (0,8)node[left]{$f_c(t)$}to(11,8);
\draw[->] (0,12)--(11,12);
```

Sinusoidal curve

```
\draw[thick] (0,12) sin (2,14) cos (4,12) sin (6,10) cos (8,12);
```

For vertical dotted axis

```
\draw[dotted] (4,16) to (4,-4);
\draw[dotted] (8,16) to (8,-4);
```

A square pulse

```
\draw[thick]
(0,9.8) to (1,9.8) to (1,6.2) to (2,6.2) to (2,9.8) to (3,9.8)
to (3,6.2) to (4,6.2) to (4,9.8) to (5,9.8) to (5,6.2) to
(6,6.2) to (6,9.8) to (7,9.8) to (7,6.2) to (8,6.2) to (8,8);
\draw
(0,9.8) node[left]{$c(t)$};
```

For dotted sinusoidal curve

```
\draw[dotted,thick]
(0,3.5) sin (2,4) cos (4,3.5) sin (6,3) cos (8,3.5);
```

An alternating signal

```
\draw[thick]
(0,3.5) sin (1,3.85) to (1,2) to (2,2) to (2,4) cos (3,3.8) to
(3,2) to (4,2) to (4,3.5) sin (5,3.2) to (5,2) to (6,2) to
(6,3.06) cos (7,3.20) to (7,2) to (8,2) to (8,3.5);
;
```

A random signal

```
\draw[dotted,thick]
(0,-2) sin (2,0) cos (4,-2) sin (6,-3) cos(8,-2);
\draw[thick]
(0,-3) to (1,-0.65) to (1.5,-0.9) to (2,0) to (3,-3) to
(4.58,-2.5) to (6,-4) to (6.8,-2.8) to (8,-4);

\end{tikzpicture}
\caption{Sample Graph}

\end{figure}
```

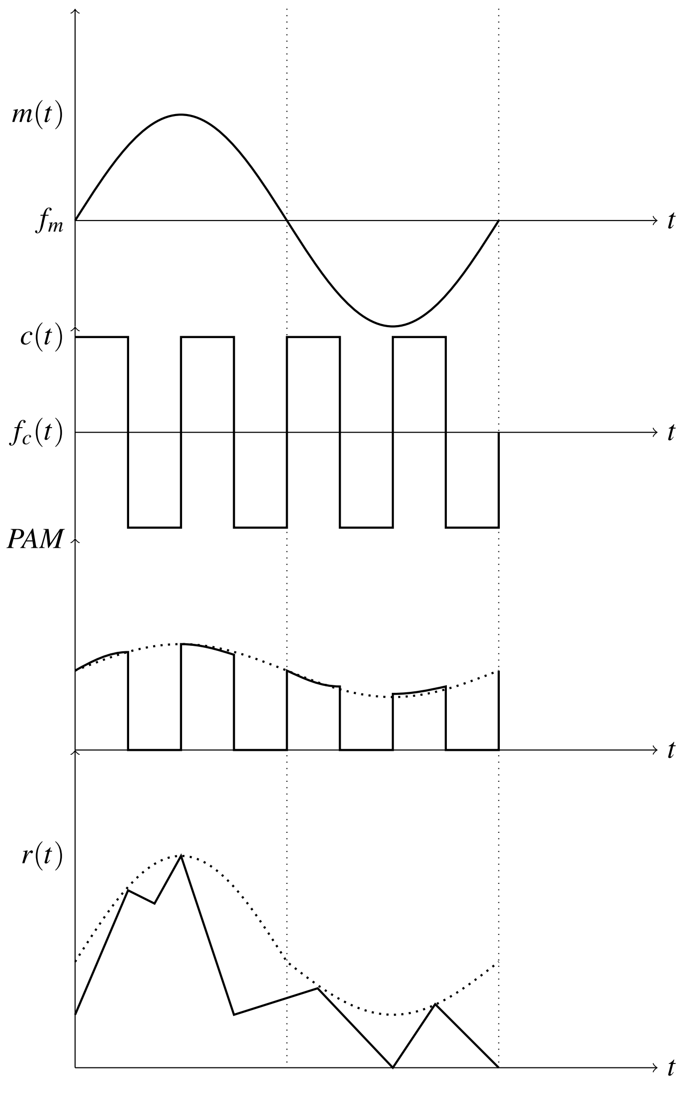


Figure 6.1: Sample Graph

7. Code Listing

LATEX is widely used in science and programming has become an important aspect in several areas of science, hence the need for a tool that properly displays code. In this article is explained how to use the standard verbatim environment as well as the package listings, which provide more advanced code-formatting features.

Displaying code in LATEX is straightforward. For instance, using the lstlisting environment:

```
\begin{lstlisting}
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
##data_logging_file=open('led.txt','w+')

while (True):
try:
GPIO.output(17,True)
##        data_logging_file.write('On\n')
time.sleep(1)
GPIO.output(17,False)
##        data_logging_file.write('Off\n')
time.sleep(1)

except KeyboardInterrupt:
GPIO.cleanup()
\end{lstlisting}
```

```
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
##data_logging_file=open('led.txt','w+')

while (True):
try:
GPIO.output(17,True)
##          data_logging_file.write('On\n')
time.sleep(1)
GPIO.output(17,False)
##          data_logging_file.write('Off\n')
time.sleep(1)

except KeyboardInterrupt:
GPIO.cleanup()
```

In this example, the outupt ignores all LATEX commands and the text is printed keeping all the line breaks and white spaces typed. To use the lstlisting environment you have to add the next line to the preamble of your document:

```
\usepackage{listings}
```

7.1 Using listings to highlight code

In the introduction a basic example of the package listings was presented, let's see a second example:

```
\begin{lstlisting}[language=Python]
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
##data_logging_file=open('led.txt','w+')

while (True):
try:
GPIO.output(17,True)
##        data_logging_file.write('On\n')
time.sleep(1)
GPIO.output(17,False)
##        data_logging_file.write('Off\n')
time.sleep(1)

except KeyboardInterrupt:
GPIO.cleanup()

\end{lstlisting}
```

The additional parameter inside brackets [language=Python] enables code highlighting for this particular programming language (Python), special words are in boldface font and comments are italicized. See the reference guide for a complete list of supported programming languages.

7.2 Importing code from a file

Code is usually stored in a source file, therefore a command that automatically pulls code from a file becomes very handy.

The next code will be directly imported from a file

```
\lstinputlisting[language=Python]{led.py}
```

```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
##data_logging_file=open('led.txt','w+')

while (True):
try:
GPIO.output(17,True)
##            data_logging_file.write('On\n')
time.sleep(1)
GPIO.output(17,False)
##            data_logging_file.write('Off\n')
time.sleep(1)

except KeyboardInterrupt:
GPIO.cleanup()
```

The command `\lstinputlisting[language=Python]{led.py}` imports the code from the file led.py, the additional parameter in between brackets enables language highlighting for the Python programming language. If you need to import only part of the file you can specify two comma-separated parameters inside the brackets. For instance, to import the code from the line 2 to the line 12, the previous command becomes

```
\lstinputlisting[language=Python, firstline=2, lastline=12]{led.py}
```

If firstline or lastline is omitted, it's assumed that the values are the beginning of the file, or the bottom of the file, respectively.

7.3 Code styles and colours

Code formatting with the listing package is highly customisable. Let's see an example

```
\documentclass{article}
\usepackage[utf8]{inputenc}

\usepackage{listings}
\usepackage{color}
```

```
\definecolor{codegreen}{rgb}{0,0.6,0}
\definecolor{codegray}{rgb}{0.5,0.5,0.5}
\definecolor{codepurple}{rgb}{0.58,0,0.82}
\definecolor{backcolour}{rgb}{0.95,0.95,0.92}

\lstdefinestyle{mystyle}{
backgroundcolor=\color{backcolour},
commentstyle=\color{codegreen},
keywordstyle=\color{magenta},
numberstyle=\tiny\color{codegray},
stringstyle=\color{codepurple},
basicstyle=\footnotesize,
breakatwhitespace=false,
breaklines=true,
captionpos=b,
keepspaces=true,
numbers=left,
numbersep=5pt,
showspaces=false,
showstringspaces=false,
showtabs=false,
tabsize=2
}

\lstset{style=mystyle}

\begin{document}
The next code will be directly imported from a file

\lstinputlisting[language=Python]{led.py}
\end{document}
```

The code colouring and styling greatly improves readability.

In this example the package color is imported and then the command `\definecolor{}{}{}` is used to define new colours in rgb format that will later be used. The package xcolor also works for this. For more information see: using colours in LATEX

There are essentially two commands that generate the style for this example:

```
\lstdefinestyle{mystyle}{...}
```

Defines a new code listing style called "mystyle". Inside the second pair of braces the options that define this style are passed; see the reference guide for a full description of these and some other parameters.

```
\lstset{style=mystyle}
```

Enables the style "mystyle". This command can be used within your document to switch to a different style if needed.

7.4 Captions and the list of Listings

Just like in floats (tables and figures), captions can be added to a listing for a more clear presentation.

```
\begin{lstlisting}[language=Python, caption=Python example]
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
##data_logging_file=open('led.txt','w+')

while (True):
try:
GPIO.output(17,True)
##        data_logging_file.write('On\n')
time.sleep(1)
GPIO.output(17,False)
##        data_logging_file.write('Off\n')
time.sleep(1)

except KeyboardInterrupt:
GPIO.cleanup()
\end{lstlisting}
```

Listing 7.1: Python example

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
##data_logging_file=open('led.txt','w+')

while (True):
try:
GPIO.output(17,True)
##        data_logging_file.write('On\n')
time.sleep(1)
GPIO.output(17,False)
##        data_logging_file.write('Off\n')
time.sleep(1)

except KeyboardInterrupt:
GPIO.cleanup()
```

Adding the comma-separated parameter caption=Python example inside the brackets, enables the caption. This caption can be later used in the list of Listings.

```
\lstlistoflistings
```

7.5 Displaying C++ codes in LATEX

In order to display the C++ codes in LATEX,the verbatim environment can be used. Example:

```
\begin{Verbatim}
#include <iostream>

using namespace std;

int main()
{
cout << "Hello World!" << endl;
return 0;
}
\end{Verbatim}
```

Hence the C++ code is displayed as:

```
#include <iostream>

using namespace std;

int main()
{
cout << "Hello World!" << endl;
return 0;
}
```

8. Circuits

8.1 Introduction

This package provides a set of macros for naturally typesetting electrical and (somewhat less naturally, perhaps) electronical networks.

`\usepackage{circuitikz}`

We don't need to load the TikZ package as well because it automatically gets loaded with circuitikz. To draw a diagram we use the circuitikz environment. We then fill the environment with a single `\draw` command ending in a **semicolon**.

8.1.1 General syntax

```
\begin{circuitikz} \draw

<circuitikz code>


;
\end{circuitikz}
```

8.2 Package Options

Different conventions can be used to represent circuit symbols. LaTeXprovides two convention for symbols used in circuits such as *currents,voltages, resistors* etc. The two available conventions in LaTeXare **american** [1] and **european.**

The package [2] to be used for american notations : `\usepackage[american]{circuitikz}`
List of options available in the respective conventions are

- **Voltages**
  - `[europeanvoltages]`: uses arrows to define voltages, and uses european-style voltage sources

[1]In all the cicuits drawn in this book, american convention is used.
[2]Loading the package without options gives european standards by default

  - `[americanvoltages]`: uses + and - to define voltages, and uses american-style voltage sources
- **Resistors**:
  - *European*: uses rectangular empty shape for resistor, as per european standards
  - *American*: uses zig-zag shape for resistors, as per american standards.
- **Inductors**
  - *European*: uses rectangular filled shape for inductor, as per european standards
  - *American*: uses ”4-bumps” shape for inductors, as per american standards
- **Ports**
  - *European*: uses rectangular logic ports, as per european standards
  - *American*: uses triangular logic ports, as per american standards

8.2.1 Other optional packages

The other optional packages which can be used along with circuitikz in the draw environment are mentioned below with brief decription of its functionality.

- *siunitx*: integrates with SIunitx package. If labels and currents are of the form `#1<#2>` (eg.: 5 mA) then what is shown actually is `\SI{#1}{#2}`(eg. `\SI{5}{\milli\ampere}`)
- *fulldiodes*: the various diodes are drawn *and* filled by default, i.e. when using styles such as diode, D, sD, ...Un-filled diode can always be forced with Do, sDo, ...
- *emptydiodes*: the various diodes are drawn but not filled by default, i.e. when using styles such as diode, D, sD, ...Filled diode can always be forced with D*,sD*, ...
- *arrowmos*: pmos and nmos have arrows analogous to those of pnp and npn transistors
- *noarrowmos*: pmos and nmos have arrows analogous to those of pnp and npn transistors
- *noarrowmos*: pmos and nmos do not have arrows analogous to those of pnp and npn transistors
- *straightlabels*: labels on bipoles are always printed straight up, i.e. with horizontal baseline
- *rotatelabels*: labels on bipoles are always printed aligned along the bipole
- *smartlabels*: labels on bipoles are rotated along the bipoles, unless the rotation is very close to multiples of 90°
- *compatibility*: makes it possibile to load CircuiTikZ and TikZ circuit library together.

8.3 Basic steps

The general format of the circuitikz code:

- A pair of co-ordinates followed by a link and then the next pair of co-ordinates.
- The link could simply be a line which is achieved using two dashes, or it could be an electrical component.
- To add a component on a line we use the keyword ‘to’ followed by square brackets containing the name of the component.
- We can then keep adding further links and co-ordinates like a chain.

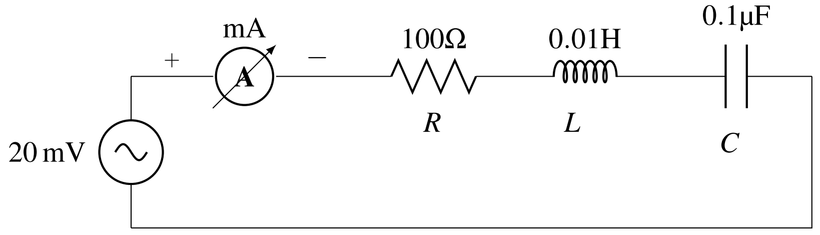
circuit maunual

8.3.1 Using actual co-ordinates

The following illustration the circuit is drawn using ‘actual co-ordinates’.

- Start the circuit with the co-ordinates (0,0)
- Between the co-ordinates (0,0) and (0,2) (i.e., vertically) a voltage source is inserted using the keyword **to**.
- From the co-ordinates (0,2) an ammeter is drawn connecting to the co-ordinate (3,2)(i.e., horizontal connection).

- Similarly
  - Resistor: (3,2) to (5,2)
  - Inductor: (5,2) to (7,2)
  - Capacitor: (7,2) to (9,2)
- The circuit can be completed in two ways:
  - By mentioning two nodes and connecting them using `--`: `(9,2)--(9,0)-- (0,0);`
  - By using a vertical dash|: `(9,2) |-(0,0)`

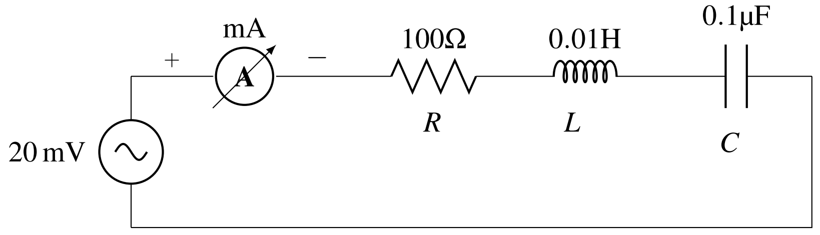


```
%using co-ordinates
\begin{circuitikz}
        \draw (0,0)
        to [ vsourcesin, l={\SI{20}{\milli\volt}}](0,2)
        to [ammeter, v^=$\si{\milli\ampere}$](3,2)
        to [R=$100\si{\ohm}$] node[pos=0.05, below left=2ex] {$R$}(5,2)
        to [L=$0.01\si{\henry}$] node[pos=0, below left=2ex] {$L$}(7,2)
        to [C=$0.1\si{\micro\farad}$] node[pos=0.75, below left=3.5ex] {$C$}(9,2)
         --(9,0)-- (0,0);
\end{circuitikz}
```

8.3.2 Using offsets

The following illustration demonstrates use of ‘offsets’ to draw the circuit.

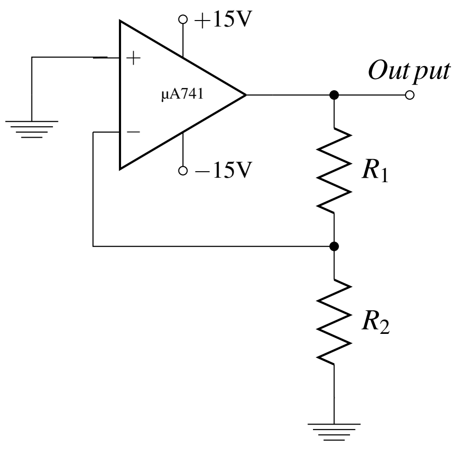
- Start the circuit with the co-ordinates (0,-5)
- The voltage source is included by adding vertical offset from the starting co-ordinates which is (0,0) using ++
- Ammeter is included by adding an horizontal offset to the previous component or link.
- Similarly
  - Resistor: `++(2,0)`
  - Inductor: `++(2,0)`
  - Capacitor: `++(2,0)`
- The circuit is completed by subtracting the offsets : `--++ (0,-2) --++ (-9,0).`



```
%using offset
\begin{circuitikz}
        \draw (0,-5)
        to [ vsourcesin,l={\SI{20}{\milli\volt}}] ++(0,2)
        to [ammeter, v^=$\si{\milli\ampere}$] ++(3,0)
        to [R=$100\si{\ohm}$] node[pos=0.05, below left=2ex] {$R$} ++(2,0)
        to [L=$0.01\si{\henry}$] node[pos=0, below left=2ex] {$L$} ++(2,0)
        to [C=$0.1\si{\micro\farad}$] node[pos=0.75, below left=3.5ex] {$C$}
        ++(2,0) --++ (0,-2) --++ (-9,0);
\end{circuitikz}
```

8.4 Non-inverting Op-amp

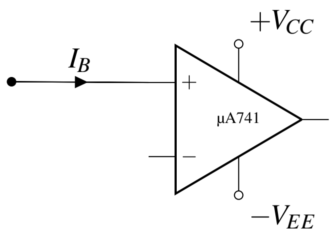
```
\begin{circuitikz}

\draw
(0,0) node[op amp,yscale=-1](o1){}
(o1.out) to (2,0)to [R,l=$R_1$,*-*](2,-2)
to [R,l=$R_2$] (2,-4)
to (2,-4)node[ground]{}
(2,0) to [short,-o](3,0)node[above]{$Output$}
;
\draw
(o1.-)|- (2,-2);
\draw
(o1.+) to (-1,0.5) to (-2,0.5) to (-2,0)node[ground]{};
% values
%op-amp parameters
\draw
(0,0)node[]{\tiny{\si{\micro\ampere}741}}
(0,0.5)|-  (0,1)node[right]{\footnotesize{$+15\si{\volt}$}} to [short,-o](0,1)
(0,-0.5)|- (0,-1)node[right]{\footnotesize{$-15\si{\volt}$}}to[short,-o](0,-1);

\end{circuitikz}
```



To draw simple non-inverting opamp circuit:

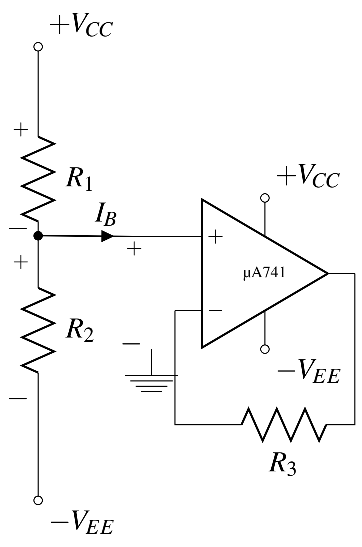
- We place an opamp at (0,0) coordinate and name it as **o1**:
  `(0,0) node[op amp,yscale=-1](o1){}`
- From the output of an opamp we draw simple line to (2,0):
  `(o1.out) to (2,0)`
- The resistors are drawn at the output terminal of the opamp,
  `(2,0)to [R,l=$R_1$,*-*](2,-2)`:
  defines the resistor with the label **R1** and by adding `*-*` will add the terminals on either side of the resistor.
- Another resistor is drawn from (2,-2) to (2,-4):
  `to [R,l=$R_2$] (2,-4)`
  at this point a ground is placed :
  `(2,-4)node[ground]{}`.
- The negative terminal of the opamp is connected to the junction of the resistors :
  `(o1.-)- (2,-2);`
- At the positive terminal of the opamp ground is connected :
  `(o1.+) to (-1,0.5) to (-2,0.5) to (-2,0)node[ground]{};`
- In the next draw we name the opamp using :
  `(0,0)node[]{\tiny{\si{\micro\ampere}741}}`
- The positive and negative voltages given to the opamp is named:
  `(0,0.5) |- (0,1)node[right]{\footnotesize{$+15\si{\volt}$}} to [short,-o](0,1)`
  `(0,-0.5)|- (0,-1)node[right]{\footnotesize{$-15\si{\volt}$}}to[short,-o](0,-1);`

8.5 Voltage Follower Opamp

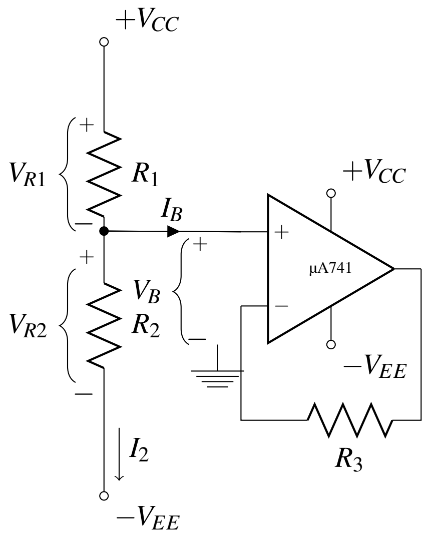


```
\begin{tikzpicture}
%opamp
\draw
(0,0)node[]{\tiny{\si{\micro\ampere}741}}
(0,0)node[op amp,yscale=-1](o1){}
(o1.+) to [short,-*,i<_=$I_B$](-3,0.5)
(0,0.5) to [short,-o](0,1)node[above right]{$+V_{CC}$}
(0,-0.5) to [short,-o](0,-1)node[below right]{$-V_{EE}$}
;
\end{tikzpicture}
```

```
\begin{tikzpicture}
%opamp
\draw
(0,0)node[]{\tiny{\si{\micro\ampere}741}}
(0,0)node[op amp,yscale=-1](o1){}
```



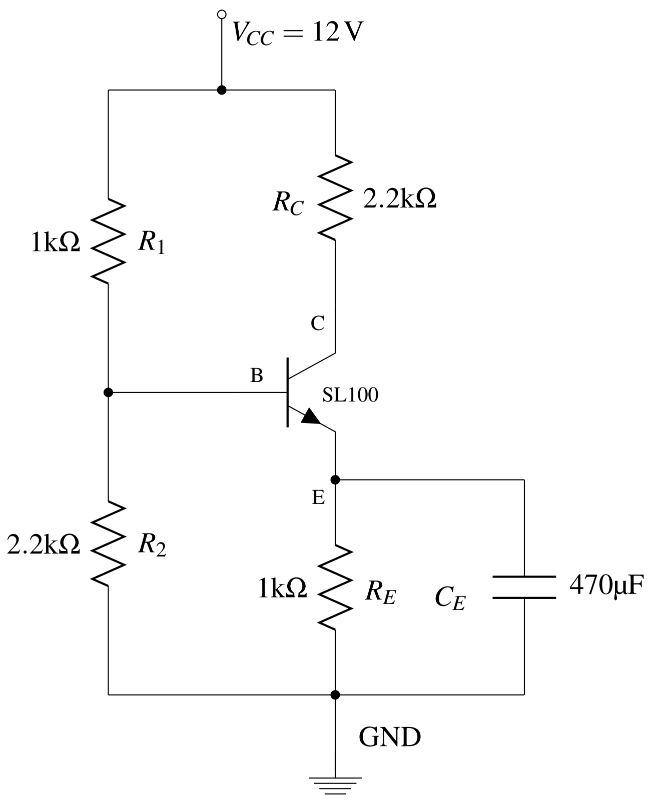
```
(o1.+) to [short,-*,i<_=$I_B$](-3,0.5)
(0,0.5) to [short,-o](0,1)node[above right]{$+V_{CC}$}
(0,-0.5) to [short,-o](0,-1)node[below right]{$-V_{EE}$}
;

%adding resistors
\draw
(-3,0.5)to [R,l=$R_2$,v_=$$](-3,-2)
to [short,-o](-3,-3)node[below right]{$-V_{EE}$}
(-3,2) to [R,l=$R_1$,v=$$](-3,0.5)
(-3,2)to [short,-o](-3,3)node[above right]{$+V_{CC}$}
(o1.-) |-(-0.5,-2) to [R,l_=$R_3$](1,-2)-| (o1.out)
(-1.5,0.4) to [open,v=$ $](-1.5,-1)node[ground]{}
;
\end{tikzpicture}
```



```
\begin{tikzpicture}
%opamp
\draw
(0,0)node[]{\tiny{\si{\micro\ampere}741}}
(0,0)node[op amp,yscale=-1](o1){}
(o1.+) to [short,-*,i<_=$I_B$](-3,0.5)
(0,0.5) to [short,-o](0,1)node[above right]{$+V_{CC}$}
(0,-0.5) to [short,-o](0,-1)node[below right]{$-V_{EE}$}
;

%adding resistors
\draw
(-3,0.5)to [R,l=$R_2$,v_=$$](-3,-2)
to [short,-o](-3,-3)node[below right]{$-V_{EE}$}
(-3,2) to [R,l=$R_1$,v=$$](-3,0.5)
(-3,2)to [short,-o](-3,3)node[above right]{$+V_{CC}$}
(o1.-) |-(-0.5,-2) to [R,l_=$R_3$](1,-2)-| (o1.out)
(-1.5,0.4) to [open,v=$ $](-1.5,-1)node[ground]{}
;

%brackets and arrows
\draw[->]
(-2.8,-2.1)node[below right]{$I_2$} to (-2.8,-2.8)
;
\draw [decorate,decoration={brace,amplitude=6pt,mirror,raise=5pt},yshift=0pt]
(-3.2,2) -- (-3.2,0.5) node [midway,xshift=-0.4cm,left] {${V_{R1}}$}
;
\draw [decorate,decoration={brace,amplitude=6pt,mirror,raise=5pt},yshift=0pt]
(-3.2,0) -- (-3.2,-1.5) node [midway,xshift=-0.4cm,left] {${V_{R2}}$}
;
\draw [decorate,decoration={brace,amplitude=6pt,mirror,raise=5pt},yshift=0pt]
(-1.7,0.4) -- (-1.7,-1) node [midway,xshift=-0.4cm,left] {${V_B}$}
;


\end{tikzpicture}
```

8.6 Emitter Folower Transistor

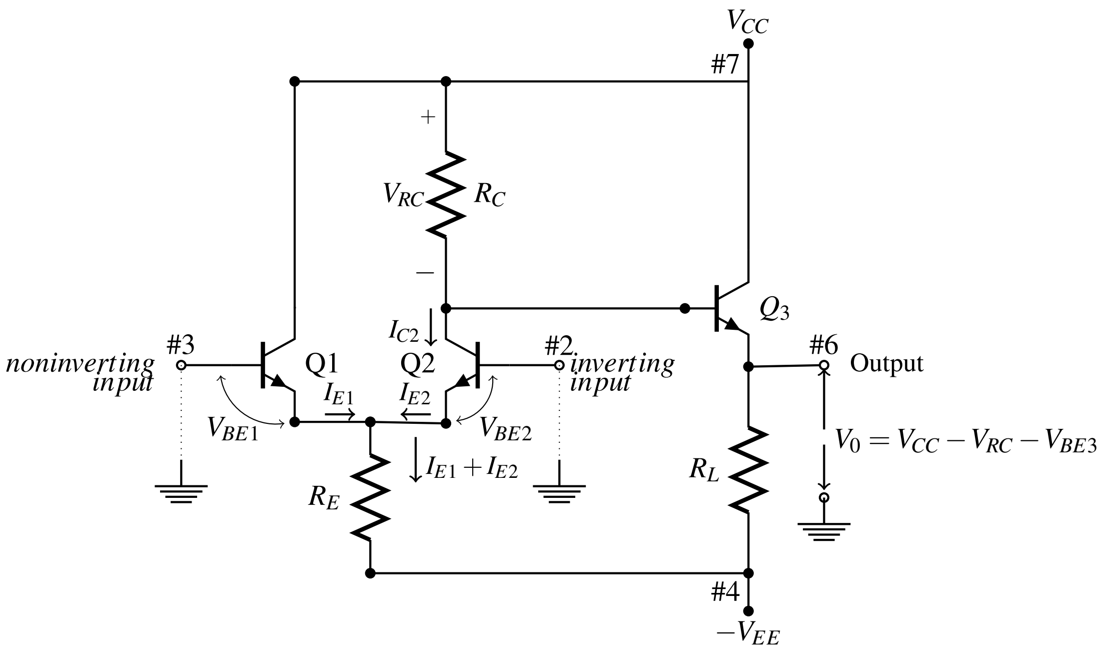
```
\begin{circuitikz}
\draw (3,4)
node[ short,*-, npn, scale=1.5](npn1) {}
(npn1.base) node[above right] {\scriptsize{B}}
(npn1.collector) node[below left] {\scriptsize{C}}
(npn1.emitter) node[below left] {\scriptsize{E}}
(npn1.emitter) node[right=0.2cm, above=0.9cm]{\scriptsize{SL100}};

\draw (0,0)
to [R=$2.2\si{\kilo\ohm}$] node[pos=-0.02, below right=1.5ex] {$R_2$} (0,4)
to [R=$1\si{\kilo\ohm}$] node[pos=-0.02, below right=1.5ex] {$R_1$}(0,8)
to (3,8)
to [R=$2.2\si{\kilo\ohm}$] node[pos=-1.2, below left=1.5ex] {$R_C$} (npn1.collector);

\draw (0,4) to [short, *-] (npn1.base);
\draw (1.5,8) to [short, *-o] node[anchor=west]{$V_{CC}=\SI{+12}{\volt}$}(1.5,9);

\draw (0,0)
to (3,0)
to [short, -*, R=$1\si{\kilo\ohm}$] node[pos=-0.09, below right=1.5ex]
 {$R_E$} (npn1.emitter)
to ++(2.5,0)
to [C=$470\si{\micro\farad}$] node[pos=-0.7, below left=3.5ex] {$C_E$} (5.5,0)
to (3,0)
to[short, *-] node[ground] {}  node[pos = 0.1, right=1ex]{GND} (3,-1);
\end{circuitikz}
```



8.7 Basic Opamp

```
\begin{tikzpicture}
\draw[color=black,thick]
(4,0) to [R,l=$R_E$,*-](4,2)
(4,2) to [short,*-*](3,2)
(3,2) to  [Tnpn,n=Q1](3,3.5)
to [short,-*](3,6.5)
to [short,-*](5,6.5)
to [R,l=$R_C$,v=$V_{RC}$](5,3.5);
\draw[color=black,thick]
(5,2.75)node[npn,xscale=-1](Q2){}
(Q2.E) to [short,*-*](4,2)
;
\draw[color=black,thick]
(9,3.5)node[npn,xscale=1](Q3){}

(Q3.B) to [short,*-*](5,3.5)
(Q3.C) to [short,-*](9,7)
(Q3.E) to [R,l_=$R_L$,*-*](9,0)
(5,6.5)--(9,6.5)
(4,0)--(9,0)
(9,0) to [short,-*](9,-0.5);



\draw[<->]
node[](A)at (2,2.75){}
node[](B) at (3,2){}
(A) to [bend right=45](B);
\draw[<->]
node[](C) at (5.6,2.75){}
node[](D) at (5,2){}
(C) to [bend left=45](D);


%Naming
\draw
(5,2.75)node[left]{Q2}
(3,2.75)node[right]{Q1}
(2.2,2.2)node[below]{$V_{BE1}$}
(5.8,2.2)node[below]{$V_{BE2}$}
(3.6,2.1)node[above]{\small$I_{E1}$}
(4.6,2.1)node[above]{\small$I_{E2}$}
(4.6,1.4)node[right]{\small$I_{E1}+I_{E2}$}
(4.8,3.2)node[left]{\small$I_{C2}$}
(9,3.5) node[right]{$Q_3$}
(9,-0.5)node[below]{$-V_{EE}$}
(8.7,-0.5)node[above]{$\#4$}
(9,7)node[above]{$V_{CC}$}
(8.7,7)node[below]{$\#7$}
;
\draw
(6.5,2.75)node[above]{$\#2$}
(6.5,2.75)node[right]{$inverting$}
(6.5,2.5)node[right]{$input$}
;
```

```
\draw[thick]
(1.5,2.75) node[above]{$\#3$}
(1.3,2.75) node[left]{$noninverting$}
(1.3,2.5) node[left]{$input$}
(Q3.E) to [short,-o](10,2.75)
(10,2.75)node[above]{$\#6$}
(10.2,2.75) to [open,-o](10,1)
(10,1)node[ground]{}
(10,1.75)node[right]{${V_0=V_{CC}-V_{RC}-V_{BE3}}$}
(10.2,2.75)node[right]{\small{Output}}
(Q2.B) to [short,-o](6.5,2.75)
(Q1.B) to [short,-o](1.5,2.75)
;
\draw[->,thick]
(3.4,2.1) to (3.8,2.1);
\draw[<-,thick]
(4.4,2.1) to (4.8,2.1);
\draw[->,thick]
(4.6,1.8) to (4.6,1.2);
\draw[->,thick]
(4.8,3.5) to (4.8,3);
\draw[<-,thick]
(10,2.7) to (10,1.9);
\draw[->,thick]
(10,1.7) to (10,1.1);


%Inputs

\draw[thick]
(6.5,1.5)node[ground]{}
(1.5,1.5)node[ground]{}
;
\draw[dotted]
(6.5,2.75) -- (6.5,1.5)
(1.5,2.75) -- (1.5,1.5)
;
\end{tikzpicture}
```


V_{CC}
#7
+
V_{RC}
R_C
−
I_{C2}
Q_3
#3
noninverting input
Q1
Q2
#2
inverting input
#6
Output
I_{E1}
I_{E2}
V_{BE1}
V_{BE2}
$I_{E1}+I_{E2}$
$V_0 = V_{CC} - V_{RC} - V_{BE3}$
R_E
R_L
#4
$-V_{EE}$

II Beamer

9. Presentation

Beamer is a LaTeX-class for creating presentations that are held using projectors. Preparing presentations in beamer is different and better than WYSIWYG(What You See Is What You Get) programs like Microsoft PowerPoint , OpenOffice , LibreOffice etc. A beamer presentation is created just like any other LaTeX.

9.1 Features

- A`\tableofcontents` will create a table of contents slide
- Overlays and dynamic effects can esily be created
- Themes to suit different presentations can be choosen
- A special style file allows you to use the LaTeX-source of a presentation directly in other LaTeXclasses like article or book. This makes it easy to create presentations out of lecture notes or lecture notes out of presentations.
- The final output is typically a pdf-file. Viewer applications for this format exist for virtually every platform. When bringing your presentation to a conference on a memory stick, you do not have to worry about which version of the presentation program might be installed there. Also, your presentation is going to look exactly the way it looked on your computer.

9.2 Structure of Beamer

- Premble
- Body
  - `\section{}`
  - `\subsection{}`
    - Slides (frames in beamer)
      - `\itemize` and `\enumerate` environments

9.2.1 Premble

The beamer package is provided with most LaTeXdistributions but is also available from CTAN.

```
\documentclass{beamer}
% Beamer class is called for the purpose of presentation in document class
\usetheme{}
 % Any presentation theme can be selected appropriate audience
```

9.2.2 Title page and information

First, you give information about authors, titles and dates in the preamble.

■ **Example 9.1** `\title[empiria] % (optional, only for long titles)`

```
{FDP on LaTeX}
\subtitle{Doing is believing }
\author[Author, Anders] % (optional, for multiple authors)
{F.~Chetan\inst{1} \and S.~Rajesh\inst{2}}
\institute[AMC Enginerring College] % (optional)
{
\inst{1}%
Institute of ECE\\
University Here
\and
\inst{2}%
Institute of ECE\\
University There
}
\date[KPT 2017] % (optional)
{Conference on Presentation Techniques, 2017}
\subject{LATEX}
```

■

Then, in the document, you add the title page :

```
\frame{\titlepage}
```

- The title for the presentation can be given using `\title{text}`
- The presenter/author can be provided using `\author{names}`
- The **title** and **author** names will **not be displayed** unless called in a frame. This can be done by using `\titlepage` in the document environment

9.2.3 Style

Themes

The first solution is to use a built-in theme such as Warsaw, Berlin, etc. The second solution is to specify colors, inner themes and outer themes.

The Built-in solution

To the preamble you can add the following line:

```
\usetheme{Warsaw}
```

to use the "Warsaw" theme. Beamer has several themes, many of which are named after cities (e.g. Frankfurt, Madrid, Berlin, etc.). This Theme Matrix 8 contains the various theme and color combinations included with beamer. For more customizing options, have a look to the official documentation included in your distribution of beamer, particularly the part *Change the way it looks*. The full list of themes is:

- Antibes
- Bergen
- Berkeley
- Berlin
- Copenhagen
- Darmstadt
- Dresden
- Frankfurt
- Goettingen
- Hannover
- Ilmenau
- JuanLesPins
- Luebeck
- Madrid
- Malmoe
- Marburg
- Montpellier
- PaloAlto
- Pittsburgh
- Rochester
- Singapore
- Szeged
- Warsaw
- boxes
- default
- CambridgeUS

Color themes, typically with animal names, can be specified with

```
\usecolortheme{beaver}
```

The full list of color themes is:

- default
- albatross
- beaver
- beetle
- crane
- dolphin
- dove
- fly
- lily
- orchid
- rose
- seagull
- seahorse
- whale
- wolverine

Fonts

You may also change the fonts for particular elements. If you wanted the title of the presentation as rendered by

```
\frame{\titlepage}
```

to occur in a serif font instead of the default sanserif, you would use:

```
\setbeamerfont{title}{family=\rm}
```

You could take this a step further if you are using OpenType fonts with Xe(La)TeX and specify a serif font with increased size and oldstyle proportional alternate number glyphs:

```
\setbeamerfont{title}{family=\rm\addfontfeatures{Scale=1.18, Numbers={Lining,
Proportional}}}
```

Math Fonts

The default settings for beamer use a different set of math fonts than one would expect from creating a simple math article. One quick fix for this is to use at the beginning of the file the option mathserif

```
\documentclass[mathserif]{beamer}
```

Others have proposed to use the command

```
\usefonttheme[onlymath]{serif}
```

but it is not clear if this works for absolutely every math character.

9.2.4 Example Code for simple beamer structure

```
\documentclass[]{beamer}
\usetheme{Dresden}

\begin{document}

\subsection{Introduction}
\subsubsection{Overview}
\begin{frame}{Why Python}
        \begin{itemize}
                \item Python is optimised for:
                \begin{itemize}
                        \item Software Quality
                        \item Developer Productivity
                        \item Program Portability
                \end{itemize}
                \item Widely used in areas:
                \begin{itemize}
                        \item Internet Scripting
                        \item System Programming
                        \item User Interfaces
                \end{itemize}
        \end{itemize}
\end{frame}

\end{document}
```

9.3 Columns and Blocks

There are two handy environments for structuring a slide: “blocks”, which divide the slide (horizontally) into headed sections, and “columns” which divides a slide (vertically) into columns. Blocks and columns can be used inside each other.

Columns

Example

```
\begin{frame}{Example of columns 1}
\begin{columns}[c] % the "c" option specifies center vertical alignment
\column{.5\textwidth} % column designated by a command
Contents of the first column
\column{.5\textwidth}
Contents split \\ into two lines
\end{columns}
\end{frame}
\begin{frame}{Example of columns 2}
\begin{columns}[T] % contents are top vertically aligned
\begin{column}[T]{5cm} % each column can also be its own environment
Contents of first column \\ split into two lines
\end{column}
\begin{column}[T]{5cm} % alternative top-align that's better for graphics
\includegraphics[height=3cm]{graphic.png}
\end{column}
\end{columns}
\end{frame}
```

Blocks

Enclosing text in the block environment creates a distinct, headed block of text (a blank heading can be used). This allows to visually distinguish parts of a slide easily. There are three basic types of block. Their formatting depends on the theme being used.

Simple

```
\begin{frame}
\begin{block}{This is a Block}
This is important information
\end{block}
\begin{alertblock}{This is an Alert block}
This is an important alert
\end{alertblock}
\begin{exampleblock}{This is an Example block}
This is an example
\end{exampleblock}
\end{frame}
```

9.4 Creating simple overlay

If it is intended to show each item sequentially then the code can be modified by adding \pause in between the points to break the flow, for explaining each point.

Items can be shown one after another, not all items right away.By entering the \pause command before every entry in a list we can reveal the list point by point.

9.4.1 Example overlay structure

```
\begin{document}
        \section{title}
        \begin{frame}
                \begin{itemize}
                \item % Frame content goes here
                \pause
                \item
                \pause
                \item
                \pause
                \item
                \end{itemize}
        \end{frame}
\end{document}
```

This structure describing overlay, divides the single frame into different slides. Every \pause gives a new slide delivering each item in each slide. This brings us on to the difference between a frame and a slide. A single frame is defined as what we build up in a single frame environment, whereas a slide is a single page in the resulting pdf. This means that a frame can be made up of multiple slides. For example, this frame with the list of four items, with each item displayed on each slide of the pdf.

An item can be emphasised by using beamer code \alert. *This* \alert *, by defaults, typesets its argument in bright red*

9.4.2 Example code for simple overlay

```
\begin{document}

\subsection{Introduction}
        \begin{frame}{Why Python}
                \begin{itemize}
                \item Python is optimised for:
                \pause
                        \begin{itemize}
                                \item Software Quality
                                \pause
                                \item Developer Productivity
                                \pause
                                \item Program Portability
                                \pause
                        \end{itemize}
                \item Widely used in areas:
                        \begin{itemize}
```

```
                                \item Internet Scripting
                                \pause
                                \item System Programming
                                \pause
                                \item User Interfaces
                        \end{itemize}
                \end{itemize}
        \end{frame}

\end{document}
```

9.5 Overlay specification

There are a number of commands that enable us to use overlays on text. The main one is the `\onslide` command which can be configured to achieve a few different outcomes.

```
\begin{frame}
        \frametitle{Overlays}
        \onslide<1->{First Line of Text}

        \onslide<2->{Second Line of Text}

        \onslide<3->{Third Line of Text}
\end{frame}
```

When we simply give this command text as an argument, it acts in the same way as the `\uncover` command making the text fully appear only on the specified slides. On unspecified slides the text is covered, so will still take up space but won't be visible.

To make the text transparent on unspecified slides we use the `\setbeamercovered` command and enter the keyword `transparent` above the code where we want it to have an effect:

```
\setbeamercovered{transparent}
```

Please be aware that this command will affect all of the code following it, so if we want to change it back to the default setting later in the presentation we can simply use the same command again but with the keyword `invisible`.

Another command we can use is the `\visible` command which does the same as `\uncover` except it leaves the space blank on unspecified slides instead of transparent even if we've set the transparency as we did a moment ago. The `\invisble` command does the exact opposite to of the `\visible` command. The `\only` command does the same as the `\visible` command except it doesn't take any space up. This means that if we change the `\onslide` commands to `\only` commands and get rid of the dashes in the overlay specifications our three lines of text will appear in the same place on the frame in turn.

```
\begin{frame}
        \frametitle{Overlays}
        \only<1>{First Line of Text}

        \only<2>{Second Line of Text}

        \only<3>{Third Line of Text}
\end{frame}
```

9.5.1 Overlays and Text Formatting

There are a number of commands to do with text formatting that are compatible with overlay specifications. These commands are simply ignored on slides not declared in the specification and will therefore just print the text as normal. Here are some examples:

```
\textbf<2>{Example Text}

\textit<2>{Example Text}

\textsl<2>{Example Text}

\textrm<2>{Example Text}

\textsf<2>{Example Text}

\textcolor<2>{orange}{Example Text}

\alert<2>{Example Text}

\structure<2>{Example Text}
```

Which will produce text like this on the first slide:

Example Text
Example Text
Example Text
Example Text
Example Text
Example Text
Example Text
Example Text

And then like this on the second slide:

Example Text
Example Text
Example Text
Example Text
Example Text
Example Text
Example Text
Example Text

First the `\textbf` command which makes the text bold, then `\textit` which puts the text in italics, then `\textsl` which make it slanted, `\textrm` which uses the roman font family, `\textsf` which uses the sans serif font family but this doesn't change anything because we are already using this font. Next the `\textcolor` command which puts it in the specified colour, then `\alert` which puts the text in red by default and finally the `\structure` command which formats the text in a way that indicates the presentations structure.

9.5.2 Overlays and Tables

We may want to animate a table so that the rows appear slide by slide. To do this we use the `\onslide` command like this. We also need to make sure we reset the `\setbeamercovered{}`

command to the default for this to work:

```
\setbeamercovered{invisible}
\begin{frame}
        \frametitle{Tables}
        \begin{table}
                \begin{tabular}{l | c | c | c | c }
                        Competitor Name & Swim & Cycle & Run & Total \\
                        \hline \hline
                        John T & 13:04 & 24:15 & 18:34 & 55:53 \onslide<2-> \\
                        Norman P & 8:00 & 22:45 & 23:02 & 53:47 \onslide<3->\\
                        Alex K & 14:00 & 28:00 & n/a & n/a \onslide<4->\\
                        Sarah H & 9:22 & 21:10 & 24:03 & 54:35
                \end{tabular}
        \caption{Triathlon results}
        \end{table}
\end{frame}
```

9.5.3 Overlay Areas

Now we combine the columns/blocks and overlays in a single frame to get a better animation.We create an overlay area in a block or a column where in the overlay specified content will appear in the same column/block.This is shown in the below example.

```
\begin{frame}{\textbf{LAMP}}
        \begin{itemize}
                \begin{columns}
                        \begin{column}{5cm}
                                \pause
                                \begin{block}{\textbf{Lamp}}
                                        \pause
                                        \item L -
                                        \textbf<3>{Linux}
                                        \item A -
                                        \textbf<4>{Apache}
                                        \item M -
                                        \textbf<5>{MySQL}
                                        \item P -
                                        \textbf<6>{PHP}
                                        \pause
                                \end{block}
                        \end{column}

                        \begin{column}{5cm}
                                \begin{overlayarea}{\textwidth}{3cm}
                                \only<3>
                                 {\includegraphics[scale=0.3]{./Images/LAMP/Linux}}
                                \only<4>
                                {\includegraphics[scale=0.1]{./Images/LAMP/Apache}}
                                \only<5>
                                {\includegraphics[scale=0.1]{./Images/LAMP/MySQL}}
                                \end{overlayarea}
                        \end{column}
                \end{columns}
        \end{itemize}
\end{frame}
```

In the above example, when the text ‘Linux’ is bold or highlighted the coresponding image will be displayed on the other column.The images are displayed on the same column corresponding to their overlay specifications.

Bibliography

[1] https://www.sharelatex.com/

[2] https://ctan.org/

[3] www.texstudio.org