LaTeX- Cheat Sheet

A collection of useful latex codes for Computer Science

Version: 1.0

Last updated: November 13, 2024

Joakim Iversen

Contents

1 Introduction

The creation of this document is for students studying Computer Science or interested in the depth of LaTeXand the possibilities therein.
The document will have different section with commands and techniques to make the layout and styling of the paper easier and more pleasant to write.

If you have suggestions for things to add, or stumble across anything that will be useful to have in the sheet. Please contact: au761308@post.au.dk

Hope you enjoy using the Cheat Sheet ☕

2 Layout

This section will have focus on some basic styling and layout in latex, such as styling text, inserting images and how to make different types of lists.

2.1 Text

Writing text in LaTeXin the most basic form is just as easy as one might expect. You just have to write in the editor. But there are some basic functions we can add to this, to make things in the text stand more out.

It is these function we will learn about in the following section.
The following section covers these usepackages and commands (Click on any part to get to the choosen subject):

Usepackage mentioned in this section:

Usepackage implementation	What the usepackage does
\usepackage{*FONT-PACKAGE-NAME*}	Used to import font for your document

2.1.1 Size

To set the font-size for your document you have to change the \documentclass{...} to include the following \documentclass[*FONT-SIZE*]{...}. When declaring the font-size you have to use the unit "pt" which stands for points (A typography unit).

Besides setting the document font-size LaTeXhas 10 different size modifiers to use inside the basic text. These modifiers are the following:

\tiny	Lorem Ipsum
\scriptsize	Lorem Ipsum
\footnotesize	Lorem Ipsum
\small	Lorem Ipsum
\normalsize	Lorem Ipsum
\large	Lorem Ipsum
\Large	Lorem Ipsum
\LARGE	Lorem Ipsum
\huge	Lorem Ipsum
\Huge	Lorem Ipsum

The sizing is gonna be active for the duration of the line you are typing. This text is therefor gonna be foot note size until i change it back using \normalsize.
It is therefor not enough to make a line change by pressing enter or using the \\line break.

Examples:

Setup:

```
\documentclass[18pt]{article}
```

Output:

```
This command sets up a document with a font size of 12pt.
```

Setup:

```
\tiny
```

Output:

```
This text is using the tiny command
```

2.1.2 Family

The font-family is the type of font you use in your LaTeXdocument. The standard font-family for a LaTeXdocument is *Computer Modern Typeface Family.* This includes different options for serif, sans serif and monospaced (Typewriter). You can switch between these throughout your document by using the following commands

\textrm	Lorem Ipsum
\textsf	Lorem Ipsum
\texttt	`Lorem Ipsum`

If you want to see more fonts you can check under the section Miscellaneous and font-family

When you have to use another font, as one shown under the section Miscellaneous, you have to import the "font package name" by implementing \usepackage{*Font-Package-Name*}. This allows you to use this font throughout your document.

If your otherwise just want to use a font in your text and not through out the whole document, you can use the code \fontfamily{*Font-Code*}\selectfont The first part of this code is to choose the font you want, where you have to insert the Font-Code as seen in the table in section 3.1.1. The second part is then sets the font, this means that whitout the \selectfont the font will not be set. If you only want the font change for a specific part of your text. you can do that the following way: {\usepackage{*Font-Package-Name*}*The Text you want to be affected by the font*}

Examples:

Setup:

\usepackage{*lmodern*}

Output:

This Commands makes your document use the Latin Modern font-family

2.1.3 Styles

There are some basic styles that we are used to having available to us in the text editors we are using on a normal basis like word and notepad. The most basic of these are *Italic*, **Bold**, and Underline.

To use these you have the following three commands

\textit	*This is Italic*
\textbf	**This is Bold**
\underline	This is underlined

If you want to find more examples of way to style your text in LaTeXyou can check under Miscellaneous and font-style

2.2 Images

The following section covers these usepackages and commands (Click on any part to get to the choosen subject):

Usepackage mentioned in this section:

Usepackage implementation	What the usepackage does
\usepackage{*graphicx*}	Used to add and manipulate images
\usepackage{wrapfig}	Used to allow text wrapping around images

When working with images there are a few basic usepackages we want to have imported at the beginning of our document.

The following list covers the basic usepackages and commands:

\usepackage{graphicx}	Allows us to insert and manipulate images in our document
\graphicspath{*PATH-FOR-IMAGE-FOLDER*}	Informs of the folder where your images is stored
\includegraphics{*NAME-OF-IMAGE*}	Inserts an image into your document

With these three commands can we insert and modify images in our document. We use the \includegraphics to import the picture:

This is the most basic way of inserting an image. And there are a few different things we can do to make the image look better.

2.2.1 Resizing and Rotating

If we want to change the size or orientation of an image we can pass in different settings in the format as shown bellow

\includegraphics[*FORMAT-SETTINGS*]{ . . . }

This means that you have to import the image into the document as normal but add the [. . .] with your modifiers before the image you're importing.

There are "three" settings we can use with our image:

rotate=XXX	Rotates the image XXX degrees
width=XXX	Sets the width of the image to XXX
height=XXX	Sets the height of the image to XXX
scale=XXX	Scales the image XXX-times

The gray box is not counted, since it has the same effect as the width, and height setting. You can combine all these settings on the same image, by separating them with a ','. The values we give to the settings can be given with a selection of different units. To see the complete list you can look under Miscellaneous and Image Units

2.2.2 Positioning

Now that we know how to insert images into our document using the \includegraphics{} command, we will look into the positioning of the image. To do this we will have to use an environment called *figure*. (We will cover environments in a later section - so just ignore that for now)

Setup:

Example
\begin{figure}[*float-value*]
\includegraphics[*setting=value*]{*IMAGE-NAME*}
\end{figure}

Entered
\begin{figure}[*H*]
\includegraphics[scale=.1]{Images/Programming.png}
\end{figure}

Output:

Here we use the *figure* environment to control the positioning of the image. The positioning is controlled by the *float-value*. There are six different parameters that we can use to control the positioning:

Parameter	Position
h	Place the float *here* - approximately where it is placed 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 float only
!	Override internal parameters LaTeXuses for determining "good" float position
H	Places the float at precisely the location in the LaTeXsource code. This requires the *float* package, and though may cause problems occasionally. This is in a way the same as using h!

Note that the '!' can be used in combination with the rest of the parameters, but not H.

2.2.3 Wrapping of text

It's also possible to wrap text around a picture. To do this we will use the "wrapfigure" environment. To be able to use this you have to implement the usepackage "wrapfig".

Setup:

\begin{wrapfigure}[LINES]{*PLACEMENT-PARRAMETER*}[OVERHANG]{*WIDTH*}
\centering
\includegraphic[*SETTING-PARAMETERS*]{*IMAGE-NAME*} \end{wrapfigure}

Output:

This is a blank text to show to possibilities of wrapping a text around an image. This text has zero actual meaning, and is just trying to fill as much space as possible.

Note that the wrapfig environment takes 4 different input parameters. The LINE parameter refers to the number of lines there should be narrowed besides the picture. In our example above we have set this to be 3, and as you can see, we have three lines that has been narrowed.

The next on is the placement of the image. Here we have 4 different values:

Position parameter	Placement of image
r	Right side of the text
l	Left side of the text
i	Inside edge-near the binding (in a *twoside* document)
o	Outside edge-far from the binding

Each of these does also have the equivalent letter in the uppercase version. When using the uppercase you allow the image to float, thus the lowercase version means to place the image *exactly* here.

The next parameter is the overhang. This is a fine-tuning of the image placement and can be used to give an image some "overhang" in the text area.

The last parameter is the width of the width of the "box" given to the image inside the environment. Because this only refers to the box given to the image, and not the image itself. It's good practice to let the image inside the wrapfig just be a tiny bit smaller than the wrapfig environment. That is how you make sure the text and image are not overlapping, and that you get a small white space between image and text.

2.2.4 Captioning, labelling and referencing

When you have inserted a figure, image or anything else into our document we want to give the reader a small explanation to what it is this figure resembles. To do this be use the code \caption{*IMAGE-CAPTION*}. We normally place the caption inside an environment with the figure we want to caption.

Setup:
\begin{figure}
\includegraphic{*IMAGE-NAME*}
\caption{Image of Computer}
\end{figure}

Output:

Figure 1: Image of Computer

As you can see we have the caption written out below the image along with *Figure 1.* The reason of the figure being numbered as number one, is because it is the first figure being captioned.

You also have a function in latex to label your figures and images. When doing this, you use the \label{*CAPTION-TITLE*} When we label our images and figures, it is not possible to see what you write in the label. But the label allows you to reference a figure later in your document. You can choose your own system for labeling, but if you don't know what to do I have inserted a table down below here, for how I in generally label my figures:

How I label	What figure
img:*IMAGE_DESCRIPTION*	Image
fig:*FIGURE_DESCRIPTION*	Figure
secc:*SECTION_DESCRIPTION*	Section
tab:*TABLE_DESCRIPTION*	Table
gra:*GRAPH_DESCRIPTION*	Graph

Table 1: Labeling Guide for Figures and Tables

By using this system I know what it is my label is referring to, whether it is a figure, image, section or table. And this is useful when you want the reference a thing from your document. I always split the label up with '_' if I use more words. This is just personal preference, and you can make a label with spaces in it if you would like.

If you want to reference a figure you have a few different options to use, each having its own unique advantages

Command	Usage
\ref{*FIG-NAME*}	Used to reference the figure number as shown below in the caption
\pageref{*FIG-NAME*}	Used to reference the page number a figure is on
\hyperref{*FIG-NAME*}{*TEXT-TO-BE-SHOWN*}	Used to create a hyperlink between placement of the reference and the figure

2.3 Environments

2.4 Lists

In LaTeXit's possible for us to create lists of different styles. Whether you want to create a bullet list, numbered list of something completely different we use these three basic environments:

- itemize environment → creating a bullet list (unordered)
- enumerate environment → creating a numbered list (ordered)
- description environment → creating a list of descriptions

The list is a powerful tool, since there are many ways for you to configure and customize how they are set up. For that reason this section is going to be quite large.

2.4.1 The basic list

The *Itemize* environment

The basic list you can create is a bullet list. To create a bullet list you will use the itemize environment. The itemize takes the same entry as all the other lists, the \item command.

The itemize list will be an unordered bullet list an would be implemented in the following way

Input:

\begin{itemize}
\item All list entries must start with \item
\item Each entry get's it's own bullet
\end{itemize}

Output:

- All list entries must start with \item
- Each entry get's it's own bullet

The *enumerate* environment

The numerate environment is used to make a ordered numbered list.

It uses the same entry command as in the itemize environment \item. But this will proceed to make a number that increases for each newly added entry to the list.

The standard enumerate environment will start at the number 1, but note that this can be changed/controlled by the use of the *enumitem package.*

Input:

\begin{enumerate}
\item Items are automatically numbered
\item Each use of the enumerate environment will make a new list with numbers starting from 1
\end{enumerate}

Output:

1. Items are automatically numbered
2. Each use of the enumerate environment will make a new list with numbers starting from 1

The *description* environment

The description environment does, as the two other, make use of the \item command to make the points in the list. The difference here lays in that you after the *item* command in '[ ]' can write a label to be put in front of the text at that point.
The syntax for this looks like this:

\item[label-text] Text of your description goes here.

You will be able to see an example here:

Input:

\begin{description}
\item This is en entry <u>WITHOUT</u> a label
\item[Something short] A short one-line description
\item[Somehting long] This is a much longer description for what is possible with a description environment. This is mostly so you can see, what it would look like if you were to have a very long description within your list.
\end{description}

Output:

This is an entry <u>WITHOUT</u> a label

Something short A short one-line description

Something long This is a much longer description for what is possible with a description environment. This is mostly so you can see, what it would look like if you were to have a very long description within your list.

2.4.2 Changing the label of lists

By using the label from the description list, we now have opened up for the possibility of changing the label of a numbered list or the bullet list.

This is done by adding the a parameter to the \item inside the [ ].

\item[*LABEL*]

An example of different ways to use this

Input:
\begin{itemize}
\item This is a normal point
\item This is a second normal point
\item[] This is an empty label
\item[!] You can use symbols from your keyboard
\item[\blacksquare] You can also use symbols
\item[NOTE] This has no bullet \end{enumerate}

Output:

- This is a normal point
- This is a second normal point

  This is an empty label

! You can use symbols from your keyboard

■ You can also use symbols

NOTE This has no bullet

This can also be done in the enumerate environment

Input:
\begin{enumerate}
\item This is a normal point
\item This is a second normal point
\item[] This is an empty label
\item[!] You can use symbols from your keyboard
\item[\blacksquare] You can also use symbols
\item[NOTE] This has no bullet \end{enumerate}

Output:

1. This is a normal point
2. This is a second normal point

   This is an empty label

! You can use symbols from your keyboard

■ You can also use symbols

NOTE This has no bullet

2.4.3 Nested lists

It's possible to nest a list inside a list. Doing this will automatically updated how the "bullet" is written, depending on which environment you are using and the level of nesting you are at

The different levels for itemize:

Input:

```
\begin{itemize/enumerate}
\item Level 1
    \begin{itemize/enumerate}
    \item Level 2
        \begin{itemize/enumerate}
        \item Level 3
            \begin{itemize/enumerate}
            \item Level 4
            \end{itemize/enumerate}
        \end{itemize/enumerate}
    \end{itemize/enumerate}
\item Back to Level 1
\end{itemize/enumerate}
```

Output:

- Level 1
  – Level 2
    ∗ Level 3
      · Level 4
- Back to Level 1

For an enumerate list it would look like this:

1. Level 1

   (a) Level 2

      i. Level 3

         A. Level 4

2. Back to Level 1

Note that we haven't been going above an nested level of 4, this is the basic max level of nested lists in LaTeX. We also haven't shown what a description list would be like, but since a description list is made to be used with the custom labels, you would also be able to create your own custom labels for the nested lists.

If you mix the lists nested inside with different lists. The lists would choose the label that matches the level of nesting it is at.

2.4.4 Customizing lists

The true power of a lists comes when you begin to modify and customize them. To do this you can make use of the *enumitem package*. This package is imported as standard and you can therefor just make use of it.

We will start by how you can customize your list labels in general.

The table bellow shows the LaTeXcommands that are being used for label-generation at each level for the *itemize* and *enumerate* lists:

Level	*Itemize* label command	*Enumerate* label command
Level 1	\labelitemi	\labelenumi
Level 2	\labelitemii	\labelenumii
Level 3	\labelitemiii	\labelenumiii
Level 4	\labelitemiv	\labelenumiv

Table 2: Table of label-generation

Besides these the *enumerate* environment also uses these four counter variables, that keep track of the current label-value for each level:

Level	*Enumerate* counter variable
Level 1	enumi
Level 2	enumii
Level 3	enumiii
Level 4	enumiv

Using the information from those two table you can begin to customize how your lists a labeled. We can make the enumerate list have numbers for all the levels of nested lists by doing this

Input:

```
\renewcommand{\labelenumii}
{\arabic{enumi}.\arabic{enumii}}

\renewcommand{\labelenumiii}
{\arabic{enumi}.\arabic{enumii}.\arabic{enumiii}}

\renewcommand{\labelenumiv}
{\arabic{enumi}.\arabic{enumii}
.\arabic{enumiii}.\arabic{enumiv}}
```

```
1. Level 1

  1.1 Level 2

    1.1.1 Level 3

      1.1.1.1 Level 4
```

You can see that the enumerate environment now labels each nested enumerate list only with numbers separated by '.' just like wrote in our *renewcommand* command. Note that the counter-variables *enumi, enumii, enumiii* and *enumiv* are counter-variables that only can be applied to the enumerate environment.

Note that in the last example we also used another command ***\arabic{}***. This explains how we want the number to be written. the \arabic{} command writes out the number using the *arabic* alphabet (0-9). There are in general five different types of formats we can print out the *counter-variable* in:

- \arabic{*Counter-Variable*}
- \roman{*Counter-Variable*}
- \Roman{*Counter-Variable*}
- \alph{*Counter-Variable*}
- \Alph{*Counter-Variable*}

If you also want your list to have more nesting than the four originals levels you can create custom counters by using the code

```
\newcounter{COUNTERTITLE}
\setcounter{COUNTERTITLE}{#}
```

So to show how the different formats look, i will write put the number five, using each format here:

- `\arabic{mycounter}` produces 5
- `\roman{mycounter}` produces v
- `\Roman{mycounter}` produces V
- `\Alph{mycounter}` produces E
- `\alph{mycounter}` produces e

the *mycounter* parameter i have given each format label is a custom counter, which you have to create for this to work, that represents the number five in my case.

To create this counter you use the following command where *COUNTER-NAME* is a local variable which you have to use when ever you call that number, and the '#' is the number you want to bind to the variable.

```
\newcouonter{COUNTER-NAME}
\setcounter{COUNTER-NAME}{#}
```

2.4.5 Customizing layout

By looking at the layout diagram shown below, we can get a look at the different parameters we can modify to change the appearance of the list we will customize.

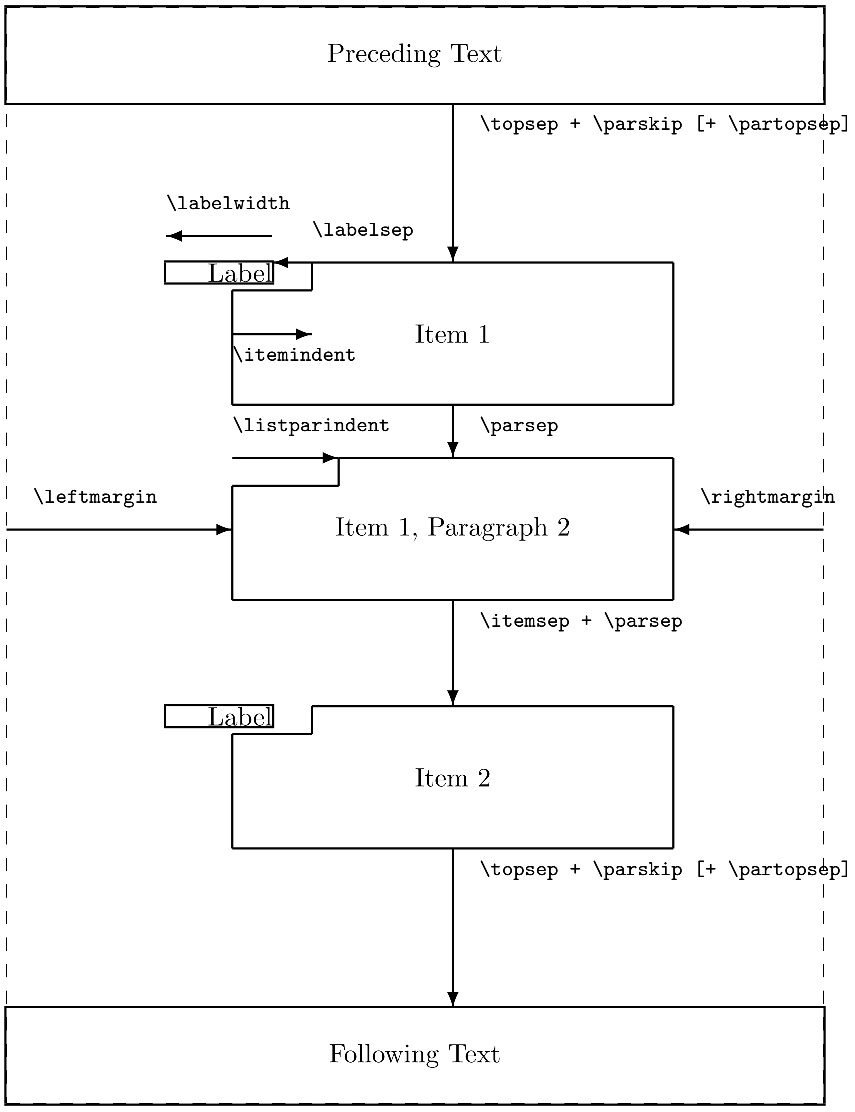

Figure 2: The LaTeX parameters which define typesetting and layout of lists.

By using these parameters we can more specific modify how we setup the list environment.

2.4.6 Custom list

In the start of the document it's possible to make a custom list environment, just like you can make a custom command.

This doesn't have to be in the top of your document, but can be done throughout. One good practice if you only us the list layout once is to do it just before the list - this way it becomes easier to see what has been done to the list. If you plan on using the custom list throughout your document, you could implement it at the top to show it has mere use.

Input:

```
\newcounter{boxblcounter}
\newcommand{\makeboxlabel}[1]{\fbox{#1.}\hfill}
\newenvironment{boxlabel}
    {\begin{list}
        {\arabic{boxblcounter}}
        {\usecounter{boxblcounter}
         \setlength{\labelwidth}{3em}
         \setlength{\labelsep}{0em}
         \setlength{\itemsep}{2pt}
         \setlength{\leftmargin}{1.5}
         \setlength{\rightmargin}{2}
         \setlength{\itemindent}{0em}
         \let\makelabel=\makeboxlabel
        }
    }
{\end{list}}
```

Output:

$\boxed{1.}$ Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.

$\boxed{2.}$ Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.

Here we have used the information on the different layout parameters in a list and the command to create a new custom environment to create our own design for a list. When you know this you can begin to go crazy and do all sort of stuff with lists

2.4.7 The power of the *enumitem* package

Using the package *enumitem* import by the following command:

Command
\usepackage{enumitem}

This package has a ot of power for manipulating lists, and we will only cover a few of the things you can do in this section.

When we have imported the *enumitem* package we can manipulate the layout of an ordinary LATEXlist by changing the parameters seen in the diagram over list layout 2. We do it by inserting '[ ]' after we initialized the environment and then just insert the parameters in the square brackets. Here we will look at an example of how we might implement this into a list. Here i will change the right-margin to be 2cm.

Input:

```
\begin{itemize}[rightmargin=2cm]
    \item \randomtext
    \begin{itemize}
        \item \randomtext
    \end{itemize}
\end{itemize}
```

Output:

- Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.
  - Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.

You can also create your own list using *enumitem* package. To do this you have to call the following command:

Command
\newlist{*name*}{*list-type*}{*max-depth*}

You have to use the three parameters: *name*, *list-type* and *max-depth*.

- *name*: the name you have to use as the environment.
- *list-type*: The type of list style you wish to copy (itemize, enumerate, description)
- *max-depth*: The max depth your list can be nested. In a standard LaTeXlist the max depth is 4.

Then when the list is created you can configure it using the \setlist command. This has the following parameters:

Command
\setlist[*NAME, DEPTH LEVEL*]{*SETTINGS*}

Here the name parameter is the name of the custom list environment you have created and the depth level is the nested level you are changing the design for. Below is given an example of a custom list with a max depth of 3, and settings for those three levels.

Input:

```
% Create List
\newlist{mylist}{enumerate}{3}

% Settings for list
\setlist[mylist, 1]{
    label=\arabic{mylisti},
    leftmargin=\parindent,
    rightmargin=10pt
}
\setlist[mylist, 2]{
    label=\arabic{mylisti}.\arabic{mylistii},
    leftmargin=15pt,
    rightmargin=15pt
}
\setlist[mylist, 3]{
    label=\arabic{mylisti}.\arabic{mylistii}.
    \arabic{mylistiii},
    leftmargin=30pt,
    rightmargin=30pt
}
```

Output:
The settings show to the left gives this list:

1 Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.

  1.1 Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.

    1.1.1 Hello, here is some text without a meaning. Hello, here is some text without a meaning. Hello, here is some text without a meaning.

You can see i am using the list counter 'mylist' which is created automatically. Using this fact its now possible for us to make a list that are nested 10+ times. This feature might not be used that much, but it's great to know.

3 Miscellaneous

3.1 Font

3.1.1 Font-Family

Font	**”Font package name”**	**”Font code”**	**Example**
Computer Modern Roman		cmr	The quick Brown Fox jumps over the lazy dog
Latin Modern Roman	lmodern	lmr	The quick Brown Fox jumps over the lazy dog
Latin Modern dunhill	lmodern	lmdh	The quick Brown Fox jumps over the lazy dog
TEXGyre Termes	tgtermes	qtm	The quick Brown Fox jumps over the lazy dog
TEXGyre Pagella	tgpagella	qpl	The quick Brown Fox jumps over the lazy dog
TEXGyre Bonum	tgbonum	qbk	The quick Brown Fox jumps over the lazy dog
TEXGyre Scholar	tgschola	qcs	The quick Brown Fox jumps over the lazy dog
Times	mathptmx	ptm	The quick Brown Fox jumps over the lazy dog
Utopia / Fourier	utopia/fourier	put	The quick Brown Fox jumps over the lazy dog
Palatino	palatino	ppl	The quick Brown Fox jumps over the lazy dog
Bookman	bookman	pbk	The quick Brown Fox jumps over the lazy dog
Charter	charter	bch	The quick Brown Fox jumps over the lazy dog
Computer Modern Sans Serif		cmss	The quick Brown Fox jumps over the lazy dog
Latin Modern Sans Serif	lmodern	lmss	The quick Brown Fox jumps over the lazy dog
TEXGyre Adventor	tgadventor	qag	The quick Brown Fox jumps over the lazy dog
TEXGyre Heros	tgheros	qhv	The quick Brown Fox jumps over the lazy dog
Helvetica	helvet	phv	The quick Brown Fox jumps over the lazy dog
Computer Modern Typewriter		cmtt	`The quick Brown Fox jumps over the lazy dog`
Latin Modern Sans Typewriter	lmodern	lmtt	`The quick Brown Fox jumps over the lazy dog`
TEXGyre Cursor	tgcursor	qcr	`The quick Brown Fox jumps over the lazy dog`
Courier	courier	pcr	`The quick Brown Fox jumps over the lazy dog`

3.1.2 Font-Styles

Style	**Command**	**Switch command**	**Output**
medium	\textmd{*INPUT*}	\mdseries	*The quick brown fox jumps over the lazy dog*
bold	\textbf{*INPUT*}	\bfseries	**The quick brown fox jumps over the lazy dog**
upright	\textup{*INPUT*}	\upshape	The quick brown fox jumps over the lazy dog
italic	\textit{*INPUT*}	\itshape	**The quick brown fox jumps over the lazy dog**
slanted	\textsl{*INPUT*}	\slshape	*The quick brown fox jumps over the lazy dog*
small caps	\textsc{*INPUT*}	\scshape	THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

3.2 Image

3.3 Image Units

pt	A point, is the default length unit approximately 0.315mm
mm	A millimeter
cm	A centimeter
in	An inch
ex	The height of an **x** in the current font
em	The height of an **m** in the current font
\columnsep	The distance between columns
\columnwidth	The Width of the column
\linewidth	The width of the line in the current environment
\paperwidth	The width of the paper
\paperheight	The height of the paper
\textwidth	The width of the text
\textheight	The height og the text
\unitlength	Units of length in the *picture* environment