Introduction to LaTeX

Liu Yihao

SJTU-UMJI Technology Department

April 15, 2020

Table of Contents

Liu Yihao

Lecture I

Hello, LaTeX

What is LaTeX

From Wikipedia, the free encyclopedia[1]

LaTeX (lah-tekh, lah-tek or lay-tek, a shortening of Lamport TeX) is a document preparation system. When writing, the writer uses plain text in markup tagging conventions to define the general structure of a document (such as article, book, and letter), to stylize text throughout a document (such as **bold** and *italic*), and to add citations[1] and cross-references.

A TeX distribution such as TeXLive or MikTeX is used to produce an output file (such as PDF or DVI) suitable for printing or digital distribution.

Within the typesetting system, its name is stylized as LaTeX.

[1] LaTeX - https://en.wikipedia.org/wiki/Latex

A brief History of TeX and LaTeX

Donald Kunuth from Stanford University is the specialist in programming art. In year 1977, he had just received his first samples from the new typesetting system of the publisher's, and its quality was so far below that of the first edition of Volume 2 that he couldn't stand it. Kunuth decided to implement a mathematical composition system by himself (since he is a computer scientist). He figured that this would take about 6 months (Ultimately, it took nearly 10 years). The system is named as TeX, of both the meaning of Greek letters $\tau\epsilon\chi$, and "technical ".

LaTeX was created in 1983 by Leslie Lamport, when he was working at SRI. He needed to write TeX macros for his own use, and thought with a little extra effort he could make a general package usable by others. Then LaTeX developed rapidly and now there are thousands of packages written in TeX macros available for direct usage.

Installation of LaTeX

Though there are some other distributions of LaTeX(like MikTeX), TeXLive is recommended in this lecture.

Windows & Linux

Download TeXLive on the tuna mirrors

https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/texlive/Images/

MacOS

Download MacTeX on the tuna mirrors

https://mirrors.tuna.tsinghua.edu.cn/CTAN/systems/mac/mactex/

Linux (Debian/Ubuntu)

Enter the command (fast with apt source mirror)

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

Selection of IDEs

There are various IDEs recommended that support LaTeX, for example

Texmaker

http://www.xm1math.net/texmaker/

Sublime Text

http://www.sublimetext.com/

Follow the instructions on https://www.zhihu.com/question/36038602

Visual Studio Code

https://code.visualstudio.com/

Follow the instructions on https://zhuanlan.zhihu.com/p/38178015

They all have cross-platform support for Windows, Linux and MacOS.

Write LaTeX on Overleaf (Online)

Another alternative choice is to write LaTeX online with the technology of Overleaf. It's free for personal usage and supports share editing which is very useful in group work.

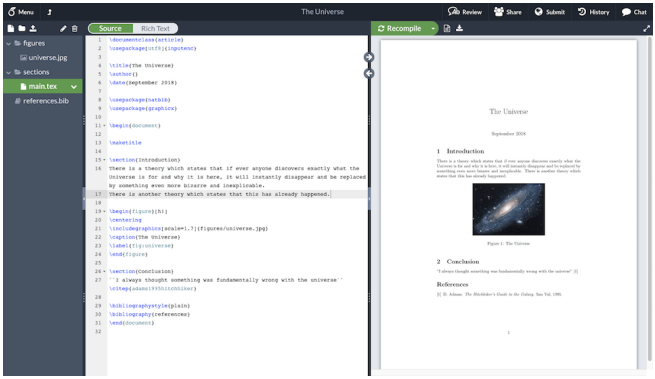


Figure 1: Layout of the Overleaf Online LaTeX Editor.

1 Getting Started
- What is LaTeX
- Distributions and IDEs
- Documentation

2 Layout of a Document

3 Learn More

Documentation of LaTeX

If you've installed a full version of TeXLive (as strongly recommended), the full LaTeX documentation is already on your computer.

Open the command line and input the command

Command

```
texdoc <docname>
```

You can also use the online version on Link

For example, you can use the following types for the docname

tex	about TeX
article	about documentclass `article`
beamer	about documentclass `beamer` (used to create slides)
pgf	about packages `tikz` and `pgf` (used to draw graphs)

Try to texdoc about all new things and then you'll be an expert in LaTeX.

A Simple Document

A typical (simplest) LaTeX example is presented here.

Example

```
\documentclass[a4paper]{article}
\usepackage{amsmath} % Define various maths environments
\usepackage{amssymb} % Define various maths symbols
\usepackage{geometry} % Adjust the margin, paper size, and etc.
\usepackage[shortlabels]{enumerate} % Provide different style of lists
\usepackage{graphicx} % Insert image of all types
% Use other packages and setup them here
\title{A simple \LaTeX\ document}
\author{XX XXX}
\date{\today}

\begin{document}
    \maketitle
    Hello, \LaTeX !
\end{document}
```

Code started with \ is called a command, and a pair of \begin{} and \begin{} is called an environment.

All Begins with documentclass

Definition

In a LaTeX file, the first line must be

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

For example, you can use the following types for the class

article	Write a report or an science article
report	Write a report
beamer	Produce a lecture silde like this!

Some options can be added, for example, a typical case can be

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

Some details about the article class will be introduced on the next page. More features about other classes and options can be found in the LaTeX Document on your own.

The article Class

The article class is one of the most basic class in LaTeX, it provides you with some normalized structure and format for report writing. So usually you will use the following command as the first line of your tex document:

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

Some of the options values are listed below (the default values are alerted)

- 10pt, 11pt, 12pt or other sizes - the font size of the document
- a4paper, a5paper, letterpaper - the size of paper
- fleqn - make the math equations left aligned (default middle aligned)
- leqno - display the serial numbers of math equations on the left (default on the right)
- titlepage, notitlepage - whether to make the title an entire page
- onecolumn, twocolumn - the number of columns of the document
- twoside, oneside - influence the position of something on the page

Other classes

This project is open sourced and you can read the source code Link to learn much (I promise) about the `beamer` class and some very interesting features of LaTeX itself. There may also be a lecture about the `beamer` class in the future.

When writing a long report, `report` class can be used to provide some more layers of document (such as `chapter`) and different type settings. It's very similar to the `article` class, so it won't be specified.

There are some other document classes such as `minimal`, `book`, `letter` and etc., but I think you may never use them.

The Preamble of a Document

As in the simple example of a document, you should notice that there is a pair of

Command

```
\begin{document}
  % some contents
\end{document}
```

This is called the body of the document, and everything before the body, including the `\documentclass` line, is called the preamble of the document.

In the preamble, you define the type of document you are writing and the language, load extra packages you will need, and set several parameters. For instance, a simplified document of the example above preamble would look like this:

Example

```
\documentclass[a4paper]{article}
\title{A simple \LaTeX\ document}
\author{XX XXX}
\date{\today}
```

Title, Author and Date

It's very useful to generate a title on the first page of a document, in order to achieve it, these commands should first be added in the preamble.

Example

```
1 \title{title}
2 \author{author name}
3 \date{\today}
```

You can simply use `\date{\today}` to display your system date now.

Then in the body (will be introduced in the next section), use the command `\maketitle` to generate the title, or title page if you added the option `titlepage` in the `\documentclass`.

Magic of Packages

LaTeX is a macro-based language, where most of useful commands are not built-in commands. These commands are defined in various packages, which should be included in the preamble.

Command

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

There are some very useful packages that you may ALWAYS include:

- `amsmath` Define various maths environments
- `amssymb` Define various maths symbols
- `geometry` Adjust the margin, paper size, and etc.
- `enumitem` Generate a list like this!
- `graphicx` Insert images of all types

The usages of these and more packages will be introduced further.

Common Packages

Here I provide a list of commonly used packages, you can start from using them after the lecture.

```
\usepackage{hyperref}    % Extensive support for hypertext
\usepackage{float}       % Improved interface for floating objects
\usepackagep[margin=2.5cm]{geometry} % Flexible document dimensions
\usepackage[shortlabels]{enumerate}  % Enumerate with redefinable labels
\usepackage{multirow}    % Tabular cells spanning multiple rows
\usepackage{multicol}    % Intermix single and multiple columns
\usepackage{ulem}        % Package for underlining
\usepackage{graphicx}    % Enhanced support for graphics
\usepackage{subfig}      % Figures broken into subfigures
\usepackage{amsmath}     % AMS mathematical facilities
\usepackage{amssymb}     % AMS symbols
\usepackage{amsfonts}    % AMS fonts
\usepackage{mathrsfs}    % Support for using RSFS fonts in maths
\usepackage{latexsym}    % LaTeX symbols
\usepackage{verbatim}    % Reimplementation of LaTeX verbatim
```

Main Body of Document

The main body of your document which starts with `\begin{document}` and ends with `\end{document}` can be also called the `document` environment. All of the contents you'd like to display should be in it, and it MUST be unique in the whole file.

Example

```
\begin{document}
    \maketitle
    \newpage   %start the following contents in a new page
    \tableofcontents %automatically generate a table of content
    \newpage
    Hello, \LaTeX !
    % TODO: Add more contents
    ...
\end{document}
```

The position and order of title page and table of contents can be arbitrary, and there can be multiple table of contents in one document.

The abstract Environment

When you are writing a paper, an abstract is often necessary in the beginning of the document.

Example

```
\begin{abstract}
  This is a lecture about how to getting start in \LaTeX!
\end{abstract}
```

Abstract

This is a lecture about how to getting start in LaTeX!

The styling of the abstract will be based on the documentclass you are using. The example shows an abstract in the beamer class, which will be slightly different from that in the article class.

Comments

As in other programming languages, comments are useful when you want to make your code readable. Adding a `%` can make the whole line after it into a comment.

Example

```
% This is a comment
```

If you need multiline comments, use the `comment` environment provided by the `comment` package. (Add `\usepackage{comment}` to your preamble.)

Example

```
\begin{comment}
 some comments
 some other comments
\end{comment}
```

Note that in the compiling, anything after a % is omitted, including the newline character, so there is no space between "comment" and "no" in the second line.

Example

```
A line
with space between ``line'' and ``with''

A line ended with comment% comments
no space between ``comment'' and ``no''
```

A line with space between "line" and "with"
A line ended with commentno space between "comment" and "no"

PS: One newline, or any number of space and tab characters are usually considered as a single "spacing" in LaTeX compilers. Two or more continuous newlines will cause a line break. We'll discuss it later in the lecture.

Sections

Commands to organize a document vary depending on the document type, the simplest form of organization is the sectioning, available in all formats.

Command

```
\section{name}
\subsection{name}
\subsubsection{name}
```

```
\section*{name}
\subsection*{name}
\subsubsection*{name}
```

The default style (can be changed with `\renewcommand`) of sections is like

1 Example Section Name
1.1 Example Subsection Name
1.1.1 Example Subsubsection Name

If an asterisk (*) is added, the sequence number will be hidden, and it won't be added to the table of contents.

Note: (Sub)sections are `commands`, and the whole contents between two (sub)sections is belonged to the former (sub)section.

Other Structures - Chapter, Part and Paragraph

Command

```
\chapter{name}
\part{name}
\paragraph{name}
\subparagraph{name}
```

```
\chapter*{name}
\part*{name}
\paragraph*{name}
\subparagraph*{name}
```

In document classes such as `report` and `book`, some outer structures of `section` (`\chapter` and `\part`) can be used.

`\paragraph` and `\subparagraph` are used for the title of small paragraphs in a (sub)section.

If an asterisk (*) is added, the effect will be the same as in the sections (sequence numbers will be hidden).

Levels of Document Structures

There are up to 7 levels of depth for defining sections depending on the document class:

Level	Command
-1	`\part{part}`
0	`\chapter{chapter}`
1	`\section{section}`
2	`\subsection{subsection}`
3	`\subsubsection{subsubsection}`
4	`\paragraph{paragraph}`
5	`\subparagraph{subparagraph}`

`\part` and `\chapter` are not available in some document classes.

Common Syntax of LaTeX Commands

All LaTeX commands have the following syntax

Command

```
\commandName<specialArgs>[optionalArgs]{requiredArgs}
```

`specialArgs` Seldom used in basic usage, for certain special usages in some packages

`optionalArgs` Used to define mode of the command, if not specified, LaTeX will use the default mode

`requiredArgs` Must be filled

If you want to connect a letter after a command, a space must be appended after the command or LaTeX won't be able to compile it correctly. But two commands can be directly connected since there is a `\` before each command.

Define New Commands

In LaTeX, you can define a new command (must not already exist) with

Command

```
\newcommand{\commandName}[args]{definition}
```

The definitions of new commands are usually put in the preamble. If there are no arguments, you can omit the optional `[args]`; or use `#num` to fill in the arguments.

Example

```
1  \newcommand{\examplelatexcommand}[1]{%
2  This lecture is #1!%
3  }%
4
5  \examplelatexcommand{interesting}
6  \examplelatexcommand{great}
```

This lecture is interesting! This lecture is great!

Here I use the comment character `%` in the end of each line of the definition to prevent adding newlines in the new command.

Renew Commands

You can also redefine a command (must already exist) with

Command

```
\renewcommand{\commandName}[args]{definition}
```

Example

```
1  \newcommand{\examplelatexcommand}[1]{...}%
2  \renewcommand{\examplelatexcommand}[1]{%
3  This lecture is not #1!%
4  }%
5
6  \examplelatexcommand{interesting}
7  \examplelatexcommand{great}
```

This lecture is not interesting! This lecture is not great!

`\renewcommand` is often used to change the style of `section`, `subsection` and etc., for example

Example

```
\renewcommand{\thesection}{\Roman{section}.}
\renewcommand{\thesubsection}{\Roman{section}.\arabic{subsection}}
```

This example changes the section number to capital roman numbers and subsection number to arabic numbers. Here's a list of available styles:

`\arabic`	prints the value as an Arabic number, e.g. 2.
`\alph`	prints the value as an alphabetic character (minuscule), e.g. b.
`\Alph`	prints the value as an alphabetic character (capital letter), e.g. B.
`\roman`	prints the value as a Roman number (minuscules), e.g. ii.
`\Roman`	prints the value as a Roman number (capital letters), e.g. II.
`\fnsymbol`	prints the value as a symbol in a sequence, this is meant to be used for symbolic footnotes, e.g. †.

Common Syntax of LaTeX Environments

All LaTeX environments have the following syntax

Command

```
\begin{environmentName}<specialArgs>[optionalArgs]{requiredArgs}
    % ...
\end{environmentName}
```

`specialArgs`, `optionalArgs`, `requiredArgs` are similar to those in a `command`

It is recommended to have a indent in each environment or your tex codes will be difficult to read by others or even yourself.

Define New Environments and Renew Environments

You can define a new environment (must not already exist) with

Command

```
\newenvironment{environmentName}[args]{before begin}{after end}
```

The difference of defining an `environment` from defining a `command` is that you should specify two code blocks, one is inserted before the `\begin` clause and the other is inserted after the `\end` clause.

Another issue is that arguments can only been used in the first of them (before `\begin`). If you need to save some arguments, use `\newcommand` to define a macro, but it may cause problems in nested usages.

Redefine an environment (must already exist) with

Command

```
\renewenvironment{environmentName}[args]{before begin}{after end}
```

For example, the examples in this lecture are provided by a self-defined `latexexample` environment:

Example

```
\newenvironment{latexexample}
{\VerbatimOut{\jobname.tmp}}
{\endVerbatimOut
\begin{example}
~
\inputminted{latex}{\jobname.tmp}

\input{\jobname.tmp}
\end{example}
}
\begin{latexexample}
  some code here
\end{latexexample}
```

It is a `verbatim` environment, which accepts LaTeX code as plain text and deals with them later.

Lecture II

Text

Use UTF-8 encoding in LaTeX

UTF-8 encoding is widely used in modern computer applications, so it's useful to include the inputenc package and use UTF-8 encoding.

Command

```
\usepackage[utf-8]{inputenc}
```

Example

café

However, different operating systems and compiling engines have different support on UTF-8 encoding, some UTF-8 codes that work on your computer may not work on others (though rarely), so it is recommended to use commands (will be introduced later) instead of directly copy and paste the UTF-8 codes from the Internet.

4 Polishing the plain text
- UTF-8 encoding
- Special symbols and accents
- Fonts
- Underline

5 Typesetting

6 Learn more - multi languages and scope in LaTeX

Special symbols

Some special symbols can't be directly used since they are reserved by LaTeX

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

Many LaTeX starters are confused with how to correctly print quotes, hyphens and dots.
` prints a left single quote, ' prints a right single quote.
`` prints a left double quote, '' prints a right double quote.
one hyphen (-) print like -
two hyphens (--) print like –
three hyphens (---) print like —
\dots prints the dots with a correct format (. . .) instead of directly use three dots (...)

Accent on letters

Sometimes you may need an accent form of a letter, here is an example of letter o

\`{o}	ò	\'{o}	ó	\^{o}	ô	\"{o}	ö	\~{o}	õ
\={o}	ō	\.{o}	ȯ	\u{o}	ŏ	\v{o}	ǒ	\H{o}	ő
\t{oo}	o͡o	\r{o}	o̊	\c{o}	o̧	\d{o}	ọ	\b{o}	o̲

Something interesting

You may be curious about how to print words like LaTeX, actually it's defined as a command.

- \TeX - TeX
- \LaTeX - LaTeX
- \LaTeXe - LaTeX 2$_\varepsilon$

Deal with unfamiliar symbols

Sometimes you may want to deal with symbols you have never seen. In this case, you may refer to `http://detexify.kirelabs.org/classify.html` to find out how to output the character.

4 Polishing the plain text
- UTF-8 encoding
- Special symbols and accents
- Fonts
- Underline

5 Typesetting

6 Learn more - multi languages and scope in LaTeX

Basic commands about fonts

First, lets start with some commands that transform font types

- \bf - **Sample Text**
- \it - *Sample Text*
- \rm - Sample Text
- \sc - Sample Text
- \sf - Sample Text
- \sl - *Sample Text*
- \tt - `Sample Text`

Note that the commands that transform font types influence the text in the whole scope ({...}) until another font type is specified. For example, how to use the first command \bf is shown below

{\bf Sample Text}

Sometimes we don't want to transform all the font types, instead, we can only change the font type of some specified text.

Example

```
\textbf{Sample text}
```

There are more options for fonts.

- \textit - *Sample Text*
- \textsc - Sample Text

However, in a math environment (will be introduced later), some other commands should be used

- \mathbf - $\mathbf{Sample\ Text}$
- \mathit - $\mathit{Sample\ Text}$
- \mathsf - $\mathsf{Sample\ Text}$

Note that the math environment doesn't include all of the font types on the previous page. More information about font types can be found here.

Font size can also be easily modified

- \tiny - Sample Text
- \scriptsize - Sample Text
- \footnotesize - Sample Text
- \small - Sample Text
- \normalsize - Sample Text
- \large - Sample Text
- \Large - Sample Text
- \LARGE - Sample Text
- \huge - Sample Text
- \Huge - Sample Text

Build a colorful document

Changing the color is similar to changing font types.

If you want to transform to a color (like transforming to bold with `\bf`), you can use `\color{name}`.

Similarly, you can use `\textcolor{name}` like `\textbf`.

The background color of the whole page can be set using `\pagecolor{name}`.

There are some defined color name in the xcolor package.

black	gray	olive	teal	blue
green	orange	violet	brown	lightgray
pink	white	cyan	lime	purple
yellow	darkgray	magenta	red	

You can find more information in the documentation of xcolor (texdoc xcolor)

4. Polishing the plain text
   - UTF-8 encoding
   - Special symbols and accents
   - Fonts
   - Underline
5. Typesetting
6. Learn more - multi languages and scope in LaTeX

Ulem package

If you want to add some lines on the text, use the ulem package.

Command

```
\usepackage{ulem}
\uline{Sample Text}
```

There are different kinds of lines supported:

- \uline - Sample Text
- \uuline - Sample Text
- \uwave - Sample Text
- \sout - ~~Sample Text~~
- \xout - Sample Text
- \dashuline - Sample Text
- \dotuline - Sample Text

Enumerate

When you need to enumerate some items as a list, you may use the enumerate package.

Command

```
1 \usepackage{enumerate}
2 \begin{enumerate}[style]
3 \item % ...
4 \item % ...
5 \item % ...
6 \end{enumerate}
```

This will generate a normal list with the serial numbers in the specified style, which could be the following (as example)

- 1 - 1, 2, 3, 4, ...
- (i) - (i), (ii), (iii), (iv), ...
- [1.] - [1.], [2.], [3.], [4.], ...

Itemize

If you want to generate an unordered list, use itemize instead of enumerate.

Command

```
\usepackage{enumerate}
\begin{itemize}
\item[style] % ...
\item[style] % ...
\item[style] % ...
\end{itemize}
```

In this case, style must be added after each item, which is different from that in enumerate, and the symbol displayed in the beginning of each item will be exactly same as the style. If style is not added, a default style will be used.

Alignment

If you want to align a paragraph of text, use these three environments for left/center/right align.

Command

```
\begin{flushleft/center/flushright}
% ...
\end{flushleft/center/flushright}
```

However, if only a single line needs to be aligned, use these three commands.

Command

```
\leftline{text}
\centerline{text}
\rightline{text}
```

Spaces may be confusing

There are defined command of spaces in different width and usages.

- ␣ - the basic space in LaTeX (printed in yellow since it's transparent). Note that any number of spaces or tabs is equal to one space, and the space after a command is ignored. If you want to add an extra space, use \␣ which makes a 1/3 em space (1 em is approximately the width of an M in the current font)
- ~ - If two words can't be separated on two lines, you can tell LaTeX about it using a tie (~), such as Prof.~Hamade (Prof. Hamade).
- \, - makes a 1/6 em space, commonly used before units (notice the space before em on this page)
- \; - makes a 2/7 em space
- \quad - makes a 1 em space
- \qquad - makes a 2 em space
- - makes actually the space of text, but text will be invisible.

Separate contents into lines and pages

Here are some basic commands about lines and pages in LaTeX, you will use them everywhere.

- \newline - begin a new line
- \\ - begin a new line (not recommended[1])
- \par - begin a new paragraph (a new line with indent)
- \\[offset] - begin a new line with an vertical offset, offset is the size of needed space (not recommended, using `\vspace` instead.)
- \newpage - begin a new page
- % - begin a line comment

[1] According to Manuel Charlemagne, \\ should only be used for a force break (where \newline doesn't work).

Spacing

When trying to separate two paragraphs by a certain space, many new learners of LaTeX may use multiple empty lines and linebreaks, which is a very dirty fix and is not so accurate. Actually, LaTeX provides a precise spacing mechanism.

Command

```
\vspace{space}
\vspace*{space}
```

When trying to show the next paragraph or sentence precisely at the bottom of the current page, we can use

Command

```
\vfill
```

between the contents of two paragraphs to separate them.

Predefined skipping

More often[1], we don't need to think about the skipping space, we can use the predefined skipping commands to achieve a small, medium or big skip. They are actually particular cases of `\vspace`

Command

`\smallskip`

`\medskip`

`\bigskip`

You may note that the effects are these skipping commands have been already shown above.

[1] According to Manuel Charlemagne, you should always use these skipping commands if possible instead of using `\\` (as in many online tutorials).

Spacing units

The space can be anything representing a size, such as 1cm, 2em and 10pt. In LaTeX, spacing units can be

- cm
- mm
- in - inch, 1 inch = 2.54 cm
- pt - 72 pt = 1 inch, the smallest unit in LaTeX
- em - 1em equals to the width of letter M
- ex - 1ex equals to the width of letter x
- `\linewidth` - the width of current line in the container
- `\pagewidth` - the width of the page
- `\pageheight` - the height of the page
- `\textwidth` - the normal width of text on the page
- `\textheight` - the normal height of text on the page

Minipage

minipage is a very useful environment for dividing pages into a grid.

Example

```
\begin{minipage}{0.32\linewidth}
  % ...
\end{minipage}
\hfill % Fill horizontal space
\begin{minipage}{0.32\linewidth}
  % ...
\end{minipage}
\hfill % Fill horizontal space
\begin{minipage}{0.32\linewidth}
  % ...
\end{minipage}
\vfill % Fill vertical space
```

```
\begin{minipage}{0.32\linewidth}
  % ...
\end{minipage}
\hfill % Fill horizontal space
\begin{minipage}{0.32\linewidth}
  % ...
\end{minipage}
\hfill % Fill horizontal space
\begin{minipage}{0.32\linewidth}
  % ...
\end{minipage}
```

The code above generate six minipages in a grid of 3 columns $\times$ 2 rows. Don't try to add up the width of minipages in a line for more than about `0.98\linewidth` (since a minipage have a small margin on each side), or the last minipage may be on a new line.

For each minipage, it can be seem as an independent LaTeX document, where text, formulas, graphics, tables and etc. can be inserted, and most importantly, they won't affect each other. What's more, you can even use minipages in a minipage to form a multi-level nesting.

The multicol package

When typesetting contents with small line width and many lines (for example, source code), the multicol package is recommended.

Command

```
\usepackage{multicol}
\begin{multicols}{cols}
    % contents on column one
    \breakcolumn % break the current column here
    % contents on column two
\end{multicols}
```

Here cols is the number of columns, it must be specified. If `\breakcolumn` is not used, the multicol package will automatically balance the length of each column.

Spelling languages

If you want to use a spelling language with characters similar to English, package babel can be used (exactly the same name as babel).

Command

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

- languages - a list of languages, the last one to be the default language

Example

```
\usepackage[greek,english]{babel}
\textgreek{abcdefgABCDEFG}
```

Then LaTeX will print αβςδεφγAB῍ΔΕΦΓ
Of course, you can use some simple commands to print these greek letters directly, such as `\alpha`, `\beta` and etc, which is more convenient only when few of them are needed.

Chinese

The Chinese TeX Community maintains a package called ctex for inputing Chinese in LaTeX. Note that it is only a package, which is shipped with most modern TeX Suites, not the CTEX Suite. I don't think it's a good choice to use the CTEX Suite directly.

Command

```
\usepackage{ctex}
```

The default LaTeX compiler pdflatex doesn't have support on Chinese input with ctex package, xelatex is a recommended modern LaTeX compiler as a replacement.

However, the ctex package is too heavy and it can slow down the total compilation speed seriously.

Usage of scope in LaTeX

First, you should realize the meaning of "scope" in programming. Let's start with a simple example in C/C++ (assuming you know that):

```
1  int main()
2  { // The scope "main" of function main
3    int a = 1; // int a is defined in scope "main"
4    for (int i = 0; i < 10; i++)
5    { // The scope "for" of the for loop
6      int b = i; // int b and i are both defined in scope "for"
7      a += b; // int a can be visited here!
8    }
9    { // The scope "other", we can directly define a scope like this
10     int c; // int c is defined in scope "other"
11     c = a; // int a can be visited here!
12   }
13   a -= c // error: c is not in scope "main", can't be visited!
14 }
```

In the example of C/C++, we use brackets {} to define a scope, which is just the same in LaTeX. In addition, notice that an environment or a command also defines a scope.

Example

```
1  black (default) text \\
2  \color{blue}
3  blue text \\
4  { \color{brown} brown text }
5  \begin{center}
6    \color{red}
7    centered red text
8  \end{center}
9  \textbf{ \color{brown}
10 bold brown text } \\
11 blue text
```

black (default) text
blue text
brown text

centered red text

bold brown text
blue text

With the usage of scopes, you can flexibly change the color, font or anything else you wish in a self-defined range of the document.

Lecture III

Maths

Introduction

Basic equations in LaTeX can be easily "programmed", for example: [1]

Example

```
1  The well known Pythagorean theorem \(x^2 + y^2 = z^2\) was
2  proved to be invalid for other exponents.
3  Meaning the next equation has no integer solutions:
4
5  \[ x^n + y^n = z^n \]
```

The well known Pythagorean theorem $x^2 + y^2 = z^2$ was proved to be invalid for other exponents. Meaning the next equation has no integer solutions:

$$x^n + y^n = z^n$$

[1] Some of this part is ported from the tutorial of Overleaf: Link

Subscripts and Superscripts

The use of superscripts and subscripts is very common in mathematical expressions involving exponents, indexes, and in some special operators. [1]

Example

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

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

Note that here we use `\[` and `\]` to typeset a mathematical expression. You may see many people (including myself in the past) using a pair of `$$` instead. It is a plain-TeX command, and is nowadays heavily deprecated. See this discussion Link on Stack Exchange for more information.

[1] Some of this part is ported from the tutorial of Overleaf: Link

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

Example

```
\[ x^{2 \alpha} - 1 = y_{ij} + y_{ij}  \]
\[ (a^n)^{r+s} = a^{nr+ns}  \]
\[ x^abc, \quad x_abc, \quad x^abc_abc \]
\[ x^{abc}, \quad x_{abc}, \quad x^{abc}_{abc} \]
```

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

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

$$x^abc, \quad x_abc, \quad x^abc_abc$$

$$x^{abc}, \quad x_{abc}, \quad x^{abc}_{abc}$$

Brackets and Parentheses

Parentheses and brackets are very common in mathematical formulas. You can easily control the size and style of brackets in LaTeX. [1]

Here's how to type some common math braces and parentheses in LaTeX:

Type	LaTeX	Code
Parentheses; round brackets	$(x+y)$	`(x+y)`
Brackets; square brackets	$[x+y]$	`[x+y]`
Braces; curly brackets	$\{x+y\}$	`\{x+y\}`
Angle brackets	$\langle x+y \rangle$	`\langle x+y \rangle`
Pipes; vertical bars	$\lvert x+y \rvert$	`x+y`
Double pipes	$\lVert x+y \rVert$	`\lVert x+y \rVert`
Floor brackets	$\lfloor x+y \rfloor$	`\lfloor x+y \rfloor`
Ceil brackets	$\lceil x+y \rceil$	`\lceil x+y \rceil`

[1] Some of this part is ported from the tutorial of Overleaf: Link

The size of brackets and parentheses can be manually set, or they can be resized dynamically in your document, as shown in the next example:

Example

```
1  \[ F = G \left( \frac{m_1 m_2}{r^2} \right) \]
```

$$F = G \left(\frac{m_1 m_2}{r^2} \right)$$

Notice that to insert the parentheses or brackets, the `\left` and `\right` commands are used. Even if you are using only one bracket, both commands are mandatory, you can use invisible brackets `\left.` or `\right.` for this.

Example

```
1  \[ \int_a^b x^2 {\rm d} x = \left. \frac{1}{3}x^3 \right|_a^b \]
```

$$\int_a^b x^2 \mathrm{d}x = \left. \frac{1}{3}x^3 \right|_a^b$$

Sometimes you may want to control the sizes of the brackets yourselves, which is called manually sized brackets. The commands listed are designed for thus purpose.

Size	LaTeX	Code
big	( )	`\big ( \big )`
Big	[ ]	`\Big [ \Big ]`
bigg	{ }	`\bigg \{ \bigg \}`
Bigg	⟨ ⟩	`\Bigg \langle \Bigg \rangle`

Mathematical Modes

LaTeX allows two writing modes for mathematical expressions: the inline mode and the display mode. The first one is used to write formulas that are part of a text. The second one is used to write expressions that are not part of a text or paragraph, and are therefore put on separate lines.

To put your equations in inline mode use `\(` and `\)`, `$` and `$` or `\begin{math}` and `\end{math}`. They all work and the choice is a matter of taste.

Example

```
1  In physics, the mass-energy equivalence is stated
2  by the equation $E=mc^2$, discovered in 1905 by Albert Einstein.
```

In physics, the mass-energy equivalence is stated by the equation $E = mc^2$, discovered in 1905 by Albert Einstein.

The display mode is usually used with mathematical environments together, which will be discussed in the next subsection.

Numbering of Equations

The display mode has two versions: numbered and unnumbered.

Example

```
1  The mass-energy equivalence is described by the famous equation
2  \[E=mc^2\]
3  discovered in 1905 by Albert Einstein.
4  In natural units ($c$ = 1), the formula expresses the identity
5  \begin{equation}
6  E=m
7  \end{equation}
```

The mass-energy equivalence is described by the famous equation

$$E = mc^2$$

discovered in 1905 by Albert Einstein. In natural units ($c = 1$), the formula expresses the identity

$$E = m \tag{1}$$

The equation Environment

An equation environment contains a set of maths equations

Command

```
\begin{equation(*)}
% ...
\end{equation(*)}
```

Example

$$\mathrm{rot}\, F = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\hat{n_x} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\hat{n_y} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\hat{n_z} \quad (2)$$

If a star(*) is added, the sequence number of the equation won't be displayed (this feature is from the amsmath package, and should behave very similar as directly using \[and \]). Note that the environment name in the \begin and \end statements must be the same (both or neither have a * here).

The LaTeX script of the equation above is quite long, but not so difficult as you think so. All of the useless spaces are omitted, so please pay attention to the necessary spaces (marked in ␣).

```
\begin{equation}
    \mathop{\rm␣rot}F=\left(\frac{\partial␣F_z}{\partial␣y}
        -\frac{\partial␣F_y}{\partial␣z}\right)\hat{n_x}
        +\left(\frac{\partial␣F_x}{\partial␣z}
        -\frac{\partial␣F_z}{\partial␣x}\right)\hat{n_y}
        +\left(\frac{\partial␣F_y}{\partial␣x}
        -\frac{\partial␣F_x}{\partial␣y}\right)\hat{n_z}
\end{equation}
```

In math environments, unlike in plain text, normal spaces will not lead to visible spaces in output. Only `\␣` or `\quad`,`\qquad` etc. will create spaces between words.

`\partial` prints the symbol ∂, `\frac{...}{...}` makes a fraction.

`\left(` and `\right(` make braces that fit the equation's height.

It is written in plain-LaTeX, and things can even be easier with packages like `physics`, which will be demonstrated later.

The split Environment (inline)

In order to deal with extremely long equations or equation with multiple lines, we can use the split environment. It is an inline environment being used in other maths environments.

Example

```
\begin{equation}
  \begin{split}
    F &= 1+2+3+4+5 \\
      &= 15
  \end{split}
\end{equation}
```

$$\begin{aligned} F &= 1+2+3+4+5 \\ &= 15 \end{aligned} \tag{3}$$

& is used to align the equal marks, and \\ is used to split the equation into two lines. Only one equation number will be generated in an equation environment.

The split environment is designed to serve as the entire body of an equation, or an entire line of an align or gather environment. There cannot be any printed material before or after it within the same enclosing structure.

The aligned Environment (inline)

For linear equation systems, the aligned environment can be used, which is similar to the split environment above. It is also an inline environment, which can be used in inline mode such as $$! Here split doesn't work because \left and \right is an enclosing structure. See this discussion Link for more information.

Example

```
Equations:
$
  \left\lbrace\begin{aligned}
    x+y &= 1 \\ x-y &= 1
  \end{aligned}\right.
  \Longrightarrow
  \left\lbrace\begin{aligned}
    x &= 1 \\ y &= 0
  \end{aligned}\right.
$
```

Equations: $\begin{cases} x+y=1 \\ x-y=1 \end{cases} \Longrightarrow \begin{cases} x=1 \\ y=0 \end{cases}$

Actually things can also be easier with packages like systeme, which will be demonstrated later.

The align Environment

An `align` environment can be used to simply the `split` or `aligned` in the `equation` environment. But it numbers the equation on each line.

Example

```
\begin{align}
  F &= 1+2+3+4+5 \\
    &= 15
\end{align}
```

$$F = 1+2+3+4+5 \tag{4}$$
$$= 15 \tag{5}$$

Use `align*` so that there will be no number(s).

Example

```
\begin{align*}
  a+b & \Leftrightarrow b+a \\
  (a+b)+c & \Leftrightarrow a+(b+c)
\end{align*}
```

$$a+b \Leftrightarrow b+a$$
$$(a+b)+c \Leftrightarrow a+(b+c)$$

The ampersand character & determines where the equations align. Let's check a more complex example:

Example

```
1  \begin{align*}
2    x&=y          &  w &=z                &  a&=b+c \\
3    2x&=-y        &  3w&=\frac{1}{2}z     &  a&=b \\
4    -4 + 5x&=2+y  &  w+2&=-1+w           &  ab&=cb
5  \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. LaTeXassumes that each equation consists of two parts separated by an &; also that each equation is separated from the one before by an &.

The gather Environment

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.

Example

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

$$2x - 5y = 8 \tag{6}$$
$$3x^2 + 9y = 3a + c \tag{7}$$

The gathered Environment (inline)

There is also an inline version of gather, called gathered. The relationship of them is similar to align and aligned.

Example

```
1  \begin{equation}
2    \begin{gathered}
3      2x - 5y =  8 \\
4      3x^2 + 9y =  3a + c
5    \end{gathered}
6  \end{equation}
```

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

The multline Environment

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.

Example

```
\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}
```

$$p(x) = 3x^6 + 14x^5y + 590x^4y^2 + 19x^3y^3 - 12x^2y^4 - 12xy^5 + 2y^6 - a^3b^3 \quad (9)$$

The equation number will be in the last line, use multline* for no numbering.

For equations equal or longer then three lines,

Example

```
\begin{multline*}
  a+b+c=1 \\
  b+c=2 \\
  c+d=1 \\
  d=3
\end{multline*}
```

$$a+b+c=1$$
$$b+c=2$$
$$c+d=1$$
$$d=3$$

Here, the first column aligns left, the last aligns right and others align center.

The flalign Environment

For equations aligned left, use the flalign environment. It is similar to the align environment.

Example

```
\begin{flalign}
  a+b &=1=& b+a \\
  b   &=2=& c
\end{flalign}
```

$$a+b=1= \qquad b+a \tag{10}$$
$$b=2= \qquad c \tag{11}$$

You may notice that the columns are aligned left except that the right most column is aligned right, different from the align environment.

Horizontal Spacing

Horizontal spacing in maths mode is useful in several situations, let's see an example: [1]

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.

[1] Some of this part is ported from the tutorial of Overleaf: Link

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

Example

```
\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*}
```

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

Vertical Spacing

When the space between display maths and the main body paragraph is considered larger than expectation, is there any way to modify the line spacing?

In default style of display mode is like

Example

```
1  your body paragraph is supposed to be typed here
2  \begin{equation}
3    a \times b = c
4  \end{equation}
5  your body paragraph is supposed to be typed here
```

your body paragraph is supposed to be typed here

$$a \times b = c \tag{12}$$

your body paragraph is supposed to be typed here

You can use \setlength to set the displayskip.

Command

```
\setlength\abovedisplayskip{<length>}
\setlength\belowdisplayskip{<length>}
```

Example

```
\setlength\abovedisplayskip{0em}
\setlength\belowdisplayskip{0em}
your body paragraph is supposed to be typed here
\begin{equation}
  a \times b = c
\end{equation}
your body paragraph is supposed to be typed here
```

your body paragraph is supposed to be typed here

$$a \times b = c \tag{13}$$

your body paragraph is supposed to be typed here

Fractions and Binomials

Fractions and binomial coefficients are common mathematical elements with similar characteristics - one number goes on top of another. [1]

Command

```
\frac{top}{bottom}  % fraction
\binom{top}{bottom} % binomial coefficients
```

Using fractions and binomial coefficients in an expression is straightforward.

Example

```
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:

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

[1] Some of this part is ported from the tutorial of Overleaf: Link

In inline and display mode, the appearance of the fractions and binomials may differ. You can use `\displaystyle` or `\textstyle` to adjust the size of the fractions and binomials, or use `\dfrac` if not all fractions in an equation need to be resized.

Example

```
1  When displaying fractions in-line, for example $\frac{3x}{2}$
2  you can set a different display style:  $ \displaystyle \frac{3x}{2} $.
3  Or you can use $\dfrac{3x}{2}$. This is also true the other way around
4  \[ f(x)=\binom{n}{x}=\frac{n!}{x!(n-x)!} \quad \textrm{and} \quad
5     f(x)=\textstyle\binom{n}{x}=\frac{n!}{x!(n-x)!} \]
```

When displaying fractions in-line, for example $\frac{3x}{2}$ you can set a different display style: $\displaystyle \frac{3x}{2}$. Or you can use $\dfrac{3x}{2}$. This is also true the other way around

$$f(x)=\binom{n}{x}=\frac{n!}{x!(n-x)!} \quad \text{and} \quad f(x)=\textstyle\binom{n}{x}=\frac{n!}{x!(n-x)!}$$

The command `\displaystyle` will format the fractions and binomials as if they were in mathematical display mode. On the other side, `\textstyle` will change the style of them as if they were part of the text.

The usage of fractions is quite flexible, they can be nested to obtain more complex expressions. And `\cfrac` can be used to make continued fractions.

Example

```
1  The fractions can be nested
2  \[ \frac{1+\frac{a}{b}}{1+\frac{1}{1+\frac{1}{a}}} \]
3  Now a wild example
4  \[ a_0+\cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cdots}}} \]
```

The fractions can be nested

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

Now a wild example

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

Operators

Characters in mathematical mode are usually shown in italics, but sometimes especial function names require different formatting (font and skip), this is accomplished by using operators defined in LaTeX. [1]

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

Example

```
\[ \sin(a + b ) = \sin(a)\cos(b) + \cos(a)\sin(b) \]
\[ \log_a b = \frac{\log_c b}{\log_c a} = \frac{\ln b}{\ln a} \]
\[ \tan a,\quad \arccos a,\quad \arcsin a,\quad \arctan a \]
```

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

$$\log_a b = \frac{\log_c b}{\log_c a} = \frac{\ln b}{\ln a}$$

$$\tan a, \quad \arccos a, \quad \arcsin a, \quad \arctan a$$

[1] Some of this part is ported from the tutorial of Overleaf: Link

Integrals

Integral expression can be added using the command

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.

Example

```
Integral $\int_{a}^{b} x^2 dx$
    ↪  inside text
\[ \int_{a}^{b} x^2 dx \]
```

Integral $\int_a^b x^2 dx$ inside text

$$\int_a^b x^2 dx$$

There is always an argue about whether *italic* or roman style of "d" should be used in integrals and derivatives. There's no right or wrong. If you prefer to use roman style, try `commath` or `physics` package. Either of them provides some macros to insert the "d" you want simply.

Multiple Integrals

To obtain double/triple/multiple integrals you must use amsmath package.

Example

```
1  \begin{gather*}
2  \iint_V \mu(u,v) \,du\,dv \\
3  \iiint_V \mu(u,v,w) \,du\,dv\,dw \\
4  \iiiint_V \mu(t,u,v,w)
   ↪  \,dt\,du\,dv\,dw \\
5  \idotsint_V \mu(u_1,\dots,u_k) \,du_1
   ↪  \dots du_k \\
6  \end{gather*}
```

$$\iint_V \mu(u,v)\,du\,dv$$
$$\iiint_V \mu(u,v,w)\,du\,dv\,dw$$
$$\iiiint_V \mu(t,u,v,w)\,dt\,du\,dv\,dw$$
$$\idotsint_V \mu(u_1,\dots,u_k)\,du_1\dots du_k$$

Cyclic Integrals

To obtain cyclic integrals you must use `esint` package.

Example

```
\begin{gather*}
\oint_V f(s) \,ds \\
\oiint_V f(s,t) \,ds\,dt \\
\end{gather*}
```

$$\oint_V f(s)\,ds$$
$$\oiint_V f(s,t)\,ds\,dt$$

Limits, Sums and Products

Like integrals, limits, sums and products expression are compressed in inline mode.

Command

```
\limits_{lower}
\sum_{lower}^{upper}
\prod_{lower}^{upper}
```

Example

```
Limit $\lim_{x\to\infty} f(x)$ inside text^^I
\[ \lim_{x\to\infty} f(x) \]
```

Limit $\lim_{x\to\infty} f(x)$ inside text

$$\lim_{x\to\infty} f(x)$$

Example

```
1  Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text
2  \[ \sum_{n=1}^{\infty} 2^{-n} = 1 \]
```

Sum $\sum_{n=1}^{\infty} 2^{-n} = 1$ inside text

$$\sum_{n=1}^{\infty} 2^{-n} = 1$$

Example

```
1  Product $\prod_{i=a}^{b} f(i)$ inside text
2  \[ \prod_{i=a}^{b} f(i) \]
```

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

$$\prod_{i=a}^{b} f(i)$$

Improvement of Integrals, Limits, Sums and Products

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 (or in display mode), use the `\limits` command before limits specification.

Example

```
1  Integral $\int_{a}^{b} x^2 dx$ inside text \par
2  Improved integral $\int\limits_{a}^{b} x^2 dx$ inside text \par
3  Use limits in display mode \[ \int\limits_{a}^{b} x^2 dx \]
```

Integral $\int_a^b x^2 dx$ inside text

Improved integral $\int\limits_a^b x^2 dx$ inside text

Use limits in display mode

$$\int\limits_a^b x^2 dx$$

Moreover, adding `\displaystyle` beforehand will make the symbol in inline mode large and easier to read, as in display mode.

Example

```
1  Limit $\lim_{x\to\infty} f(x)$ inside text \par
2  Display style limit $\displaystyle\lim_{x\to\infty} f(x)$ inside text
```

Limit $\lim_{x\to\infty} f(x)$ inside text
Display style limit $\displaystyle\lim_{x\to\infty} f(x)$ inside text

On the other hand, `\mathlarger` command (provided by `relsize` package) is used to get bigger integral symbol in display.

Example

```
1  $\int\frac{1}{2}dx - \mathlarger{ \int\frac{1}{2}dx}$
```

$\int \frac{1}{2}dx - \int \frac{1}{2}dx$

Other Math Symbols

Some examples of other common used math symbols are shown.

Name	LaTeX	Code
Square Root	$\sqrt{a}\ \sqrt[b]{a}$	`\sqrt {a}\ \sqrt [b]{a}`
Over/Under Line	$\overline{a+b}\ \underline{a+b}$	`\overline {a+b}\ \underline {a+b}`
Over Brace	$\overbrace{1+2+\cdots+n}^{n}$	`\overbrace {1+2+\cdots +n}^n`
Under Brace	$\underbrace{1+2+\cdots+n}_{n}$	`\underbrace {1+2+\cdots +n}_n`
Over Arrow	$\overrightarrow{a+b}\ \overleftarrow{a+b}$	`\overrightarrow {a+b}\ \overleftarrow {a+b}`
Under Arrow	$\underrightarrow{a+b}\ \underleftarrow{a+b}$	`\underrightarrow {a+b}\ \underleftarrow {a+b}`
Dots	$\dots \cdot \cdots \vdots \ddots$	`\dots \ \cdot \ \cdots \ \vdots \ \ddots`
Arrows	$\rightarrow \leftarrow \leftrightarrow$	`\rightarrow \ \leftarrow \ \leftrightarrow`
	$\Rightarrow \Leftarrow \Leftrightarrow$	`\Rightarrow \ \Leftarrow \ \Leftrightarrow`
	$\longleftarrow \Longrightarrow$	`\longleftarrow \ \Longrightarrow`

Mathematical Fonts

In mathematical mode as well as in text mode, you can change the typeface as needed. For instance, it's customary to represent real numbers with a blackboard bold font, or topological spaces with calligraphic font. [1]

For some elements is convenient to have the possibility of changing the font typeface.

Example

```
1  Let \( \mathcal{T} \) be a topological space, a basis is defined as
2  \[ \mathcal{B} = \{B_{\alpha} \in \mathcal{T}\, |\,
3     U = \bigcup B_{\alpha} \forall U \in \mathcal{T} \} \]
```

Let $\mathcal{T}$ be a topological space, a basis is defined as

$$\mathcal{B} = \{B_\alpha \in \mathcal{T} \,|\, U = \bigcup B_\alpha \forall U \in \mathcal{T}\}$$

[1] Some of this part is ported from the tutorial of Overleaf: Link

Mathematical Fonts for Capital Letters

There are some font typefaces that support only a limited number of characters; these fonts usually denote some special sets.

Example

```
\begin{gather*}
  RQSZ \\
  \mathcal{RQSZ} \\
  \mathfrak{RQSZ} \\
  \mathbb{RQSZ}
\end{gather*}
```

$$RQSZ$$
$$\mathcal{RQSZ}$$
$$\mathfrak{RQSZ}$$
$$\mathbb{RQSZ}$$

This example shows Calligraphic, Fraktur and Blackboard bold typefaces. For instance, to display the R in blackboard bold typeface `$\mathbb{R}$` will do the trick.

Other Mathematical Fonts

It's possible to set a different font family for a complete mathematical expression.

Example

```
\begin{gather*}
  3x^2 \in R \subset Q \\
  \mathnormal{3x^2 \in R \subset Q} \\
  \mathrm{3x^2 \in R \subset Q} \\
  \mathit{3x^2 \in R \subset Q} \\
  \mathbf{3x^2 \in R \subset Q} \\
  \mathsf{3x^2 \in R \subset Q} \\
  \mathtt{3x^2 \in R \subset Q}
\end{gather*}
```

$$3x^2 \in R \subset Q$$
$$3x^2 \in R \subset Q$$
$$3\mathrm{x}^2 \in \mathrm{R} \subset \mathrm{Q}$$
$$3x^2 \in R \subset Q$$
$$\mathbf{3x^2 \in R \subset Q}$$
$$3\mathsf{x}^2 \in \mathsf{R} \subset \mathsf{Q}$$
$$3\mathtt{x}^2 \in \mathtt{R} \subset \mathtt{Q}$$

In this case, not only letters but all characters change its appearance, for example `$\mathit{3x^2}$` italicises the entire expression.

Define Own Symbols

If you need to add a personalized operator to be displayed in Roman font instead of italics use `\DeclareMathOperator`, provided by the the package `amsmath`.

Example

```
\DeclareMathOperator{\Mr}{M_{\mathbb{R}}}
User-defined operator for matrices with Real entries $ x \in \Mr $
```

User-defined operator for matrices with Real entries $x \in \mathrm{M}_{\mathbb{R}}$

The command can be slightly modified if you need that your defined operator uses subscripts, as the `\lim` operator, in such case use `\DeclareMathOperator*`.

You can also use `\mathop` to define a italics math operator supporting subscripts, and change it to Roman font by hand.

Example

```
\[ \mathop{\mathrm{limsup}}_{n\to\infty}\mathop{rot}F_n \]
```

$$\operatorname*{\mathrm{limsup}}_{n\to\infty} \mathop{rot} F_n$$

The matrix Environment (inline)

There are various kinds of matrix environments defined in amsmath package, they are matrix, pmatrix, bmatrix, Bmatrix, vmatrix, Vmatrix.

Command

```
\begin{[p/b/B/v/V]matrix}
  a_{11} & a_{12} & ... & a_{1n} \\
  a_{21} & a_{22} & ... & a_{2n} \\
  ...    & ...    & ... & ...    \\
  a_{n1} & a_{n2} & ... & a_{nn} \\
\end{[p/b/B/v/V]matrix}
```

Example

```
\begin{equation}
  \begin{pmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33} \\
  \end{pmatrix}
\end{equation}
```

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{14}$$

Here is some examples of the style of these matrix.

Example

`matrix`

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

`bmatrix`

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

`vmatrix`

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

`pmatrix`

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

`Bmatrix`

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

`Vmatrix`

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

Some packages may also help simplify the typesetting of matrix, for example, there is some macros defined in the `physics` package to make identity matrix, or generate the examples above more simply.

If you need to create matrices with different delimiters, you can add them manually to a plain matrix. For example:

Example

```
\begin{equation}
  \left\lceil
  \begin{matrix}
    1 & 2 & 3 \\
    a & b & c
    \end{matrix}
  \right\rceil
\end{equation}

\begin{equation}
  \left\langle
  \begin{matrix}
    1 & 2 & 3 \\
    a & b & c
  \end{matrix}
  \right\rangle
\end{equation}
```

$$\left\lceil \begin{matrix} 1 & 2 & 3 \\ a & b & c \end{matrix} \right\rceil \tag{15}$$

$$\left\langle \begin{matrix} 1 & 2 & 3 \\ a & b & c \end{matrix} \right\rangle \tag{16}$$

The smallmatrix Environment

When typesetting inline math, the usual matrix environments above may look too big. It may be better to use smallmatrix in such situations, although you will need to provide your own delimiters.

Example

```
1  Trying to typeset an inline matrix here
2  $\begin{pmatrix}
3    a & b \\
4    c & d
5  \end{pmatrix}$
6  but it looks too big, so let's try
7  $\big(\begin{smallmatrix}
8    a & b \\
9    c & d
10 \end{smallmatrix}\big)$ instead.
```

Trying to typeset an inline matrix here $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ but it looks too big, so let's try $\big(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\big)$ instead.

The array Environment

An array environment is actually a math mode tabular environment, and the usage of them are almost the same. You can refer to the lecture about tables for this part.

A simple example is given here:

Example

```
\begin{equation}
  \chi(\lambda) =
  \left| \begin{array}{ccc}
    \lambda - a & -b & -c \\
    -d & \lambda - e & -f \\
    -g & -h & \lambda - i
  \end{array} \right|
\end{equation}
```

$$\chi(\lambda) = \begin{vmatrix} \lambda - a & -b & -c \\ -d & \lambda - e & -f \\ -g & -h & \lambda - i \end{vmatrix} \tag{17}$$

The AMS-LaTeX Packages

AMS-LaTeX is a collection of LaTeX document classes and packages developed for the American Mathematical Society (AMS).

It is an extension of plain-LaTeX maths, with many new maths environments (most of them were introduced in the previous section), maths symbols and maths fonts.

Usually you can insert all of the commands in the preamble of your document.

Command

```
\usepackage{amsmath}  % loads maths environments
\usepackage{amssymb}  % loads maths symbols
\usepackage{amsfonts} % loads maths fonts
```

Some Other Packages

Recall that we also use some other packages in this lecture:

Command

```
\usepackage{esint}    % for cyclic integrals
\usepackage{relsize}  % for \mathlarger
```

For a better `array` environment, though it's not mandatory (you can use it without the package), you're recommended to add the `array` package.

Command

```
\usepackage{array}
```

7 Use Maths in LaTeX

8 Useful Maths Packages
- Common Packages
- The physics Package
- The systeme Package
- The gauss Package

The physics Package

To use the physics package, simply insert the command in the preamble of your document.

Command

```
1  \usepackage{physics}
```

The goal of this package is to make typesetting equations for physics simpler, faster, and more human-readable. But it can also be used in various maths circumstances.

To that end, the commands included in this package have names that make the purpose of each command immediately obvious and remove any ambiguity while reading and editing physics code.

The documentation of the physics package can be found in http://mirrors.ctan.org/macros/latex/contrib/physics/physics.pdf.

Recall the equation:

```
\begin{equation}
    \mathop{\rm rot}F=\left(\frac{\partial F_z}{\partial y}
        -\frac{\partial F_y}{\partial z}\right)\hat{n_x}
        +\left(\frac{\partial F_x}{\partial z}
        -\frac{\partial F_z}{\partial x}\right)\hat{n_y}
        +\left(\frac{\partial F_y}{\partial x}
        -\frac{\partial F_x}{\partial y}\right)\hat{n_z}
\end{equation}
```

Now we can rewrite it with the commands defined in the `physics` package.

Example

```
\begin{equation}
  \mathop{\rm rot}F=\qty(\pdv{F_z}{y}-\pdv{F_y}{z})\hat{n_x}+
                    \qty(\pdv{F_x}{z}-\pdv{F_z}{x})\hat{n_y}+
                    \qty(\pdv{F_y}{x}-\pdv{F_x}{y})\hat{n_z}
\end{equation}
```

$$\mathrm{rot}\, F = \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\hat{n_x} + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\hat{n_y} + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\hat{n_z} \tag{18}$$

Automatic Bracing

When typesetting maths equations, you may use something like `\left(` and `\right)` to make the braces taller than the typical ones. The `physics` package provides some macros to simplify and replace them.

`\quantity`

`\qty(\typical)` → (■)
`\qty(\tall)` → (■)
`\qty(\grande)` → (■)
automatic ( ) braces

`\qty[\typical]` → [■]
automatic [ ] braces

`\qty|\typical|` → |■|
automatic | | braces

`\qty{\typical}` → {■}
automatic { } braces

`\qty\big{}` → { }
`\qty\Big{}` → { }
`\qty\bigg{}` → { }
`\qty\Bigg{}` → { }
manual sizing (works with any of the above bracket types)

`\pqty{}` ↔ `\qty()`
`\bqty{}` ↔ `\qty[]`
`\vqty{}` ↔ `\qty||`
`\Bqty{}` ↔ `\qty{}`
alternative syntax; robust and more LaTeX-friendly

`\absolutevalue`	`\abs{a}` → $\abs{a}$	automatic sizing; equivalent to `\qty |a|`
	`\abs\Big{a}` → $\abs\Big{a}$	inherits manual sizing syntax from `\qty`
	`\abs*{\grande}` → $\abs*{\grande}$	star for no resize
`\norm`	`\norm{a}` → $\norm{a}$	automatic sizing
	`\norm\Big{a}` → $\norm\Big{a}$	manual sizing
	`\norm*{\grande}` → $\norm*{\grande}$	star for no resize
`\evaluated`	`\eval{x}_0^\infty` → $\eval{x}_0^\infty$	vertical bar for evaluation limits
	`\eval(x|_0^\infty` → $\eval(x\|_0^\infty$	alternate form
	`\eval[x|_0^\infty` → $\eval[x\|_0^\infty$	alternate form
	`\eval[\venti|_0^\infty` → $\eval[\venti\|_0^\infty$	automatic sizing
	`\eval*[\venti|_0^\infty` → $\eval*[\venti\|_0^\infty$	star for no resize
`\order`	`\order{x^2}` → $\order{x^2}$	order symbol; automatic sizing and space handling
	`\order\Big{x^2}` → $\order\Big{x^2}$	manual sizing
	`\order*{\grande}` → $\order*{\grande}$	star for no resize

Vector Notation

You may use `\mathbf` to make bold maths symbols, However, it won't always work. For example, with `\mathbf{\alpha}` you may have α, which is actually not bold. These commands will help provide the correct $\boldsymbol{\alpha}$.

`\vectorbold`	`\vb{a}` → $\mathbf{a}$	upright/no Greek
	`\vb*{a}`, `\vb*{\theta}` → $\boldsymbol{a}$, $\boldsymbol{\theta}$	italic/Greek
`\vectorarrow`	`\va{a}` → $\vec{\mathbf{a}}$	upright/no Greek
	`\va*{a}`, `\va*{\theta}` → $\vec{\boldsymbol{a}}$, $\vec{\boldsymbol{\theta}}$	italic/Greek
`\vectorunit`	`\vu{a}` → $\hat{\mathbf{a}}$	upright/no Greek
	`\vu*{a}`, `\vu*{\theta}` → $\hat{\boldsymbol{a}}$, $\hat{\boldsymbol{\theta}}$	italic/Greek

There are also some shorthand for vector operations.

`\dotproduct`	`\vdot` → $\cdot$ as in $\mathbf{a} \cdot \mathbf{b}$	note: `\dp` is a protected TeX primitive
`\crossproduct`	`\cross` → $\times$ as in $\mathbf{a} \times \mathbf{b}$	alternate name
	`\cp` → $\times$ as in $\mathbf{a} \times \mathbf{b}$	shorthand name

The default del (nabla) symbol ∇ used in `physics` vector notation can be switched to appear with an arrow $\vec{\nabla}$ by including the option `arrowdel` in the document preamble → `\usepackage[arrowdel]{physics}`.

`\divergence`

`\div` → $\nabla\cdot$

`\div{\vb{a}}` → $\nabla \cdot \mathbf{a}$ — default mode

`\div(\vb{a}+\tall)` → $\nabla \cdot \left(\mathbf{a} + \blacksquare\right)$ — long-form

`\div[\vb{a}+\tall]` → $\nabla \cdot \left[\mathbf{a} + \blacksquare\right]$

note: `amsmath` symbol ÷ renamed `\divisionsymbol`

`\curl`

`\curl` → $\nabla\times$

`\curl{\vb{a}}` → $\nabla \times \mathbf{a}$ — default mode

`\curl(\vb{a}+\tall)` → $\nabla \times \left(\mathbf{a} + \blacksquare\right)$ — long-form

`\curl[\vb{a}+\tall]` → $\nabla \times \left[\mathbf{a} + \blacksquare\right]$

`\laplacian`

`\laplacian` → ∇^2

`\laplacian{\Psi}` → $\nabla^2 \Psi$ — default mode

`\laplacian(\Psi+\tall)` → $\nabla^2 \left(\Psi + \blacksquare\right)$ — long-form

`\laplacian[\Psi+\tall]` → $\nabla^2 \left[\Psi + \blacksquare\right]$

Operators

The standard set of trig functions is redefined in `physics` to provide automatic braces that behave like `\qty()`. In addition, an optional power argument is provided. This behavior can be switched off by including the option `notrig` in the preamble → `\usepackage[notrig]{physics}`.

For example,

`\sin`	`\sin(\grande)` → $\sin\left(\blacksquare\right)$	automatic braces; old `\sin` renamed `\sine`
	`\sin[2](x)` → $\sin^2(x)$	optional power
	`\sin x` → $\sin x$	can still use without an argument

Similar behavior has also been extended to the following functions:

`\exp(\tall)`	$\exp\left(\blacksquare\right)$		`\exponential`
`\log(\tall)`	$\log\left(\blacksquare\right)$		`\logarithm`
`\ln(\tall)`	$\ln\left(\blacksquare\right)$	old definitions ⇒	`\naturallogarithm`
`\det(\tall)`	$\det\left(\blacksquare\right)$		`\determinant`
`\Pr(\tall)`	$\Pr\left(\blacksquare\right)$		`\Probability`

There are also some new operators:

`\trace` or `\tr`	`\tr\rho` → $\tr\rho$ also `\tr(\tall)` → $\tr\left(\blacksquare\right)$	trace; same bracing as trig functions
`\Trace` or `\Tr`	`\Tr\rho` → $\Tr\rho$	alternate
`\rank`	`\rank M` → $\rank M$	matrix rank
`\erf`	`\erf(x)` → $\erf(x)$	Gauss error function
`\Res`	`\Res[f(z)]` → $\Res[f(z)]$	residue; same bracing as trig functions
`\principalvalue`	`\pv{\int f(z) \dd{z}}` → $\pv{\int f(z)\dd{z}}$ `\PV{\int f(z) \dd{z}}` → $\PV{\int f(z)\dd{z}}$	Cauchy principal value alternate
`\Re`	`\Re{z}` → $\Re{z}$	old `\Re` renamed to `\real` → $\real$
`\Im`	`\Im{z}` → $\Im{z}$	old `\Im` renamed to `\imaginary` → $\imaginary$

Quick Quad Text

This set of commands produces text in math-mode padded by \quad spacing on either side. This is meant to provide a quick way to insert simple words or phrases in a sequence of equations. Each of the following commands includes a starred version which pads the text only on the right side with \quad for use in aligned environments such as cases.

\qqtext	\qq{}	general quick quad text with argument
	\qq{word or phrase} → word or phrase	normal mode; left and right \quad
	\qq*{word or phrase} → word or phrase	starred mode; right \quad only

Some special macros:

\qcomma or \qc →,	right \quad only
\qcc → c.c.	complex conjugate; left and right \quad unless starred \qcc* → c.c.
\qif → if	left and right \quad unless starred \qif* → if

\qthen, \qelse, \qotherwise, \qunless, \qgiven, \qusing, \qassume, \qsince, \qlet, \qfor, \qall, \qeven, \qodd, \qinteger, \qand, \qor, \qas, \qin

Derivatives

The default differential symbol d which is used in `\differential` and `\derivative` can be switched to an italic form d by including the option `italicdiff` in the preamble → `\usepackage[italicdiff]{physics}`.

`\differential`	`\dd` → d	
	`\dd x` → $\mathrm{d}x$	no spacing (not recommended)
	`\dd{x}` → $\,\mathrm{d}x\,$	automatic spacing based on neighbors
	`\dd[3]{x}` → d^3x	optional power
	`\dd(\cos\theta)` → $\mathrm{d}(\cos\theta)$	long-form; automatic braces
`\derivative`	`\dv{x}` → $\dfrac{\mathrm{d}}{\mathrm{d}x}$	one argument
	`\dv{f}{x}` → $\dfrac{\mathrm{d}f}{\mathrm{d}x}$	two arguments
	`\dv[n]{f}{x}` → $\dfrac{\mathrm{d}^n f}{\mathrm{d}x^n}$	optional power
	`\dv{x}(\grande)` → $\dfrac{\mathrm{d}}{\mathrm{d}x}\left(\blacksquare\right)$	long-form; automatic braces, spacing
	`\dv*{f}{x}` → $\mathrm{d}f/\mathrm{d}x$	inline form using `\flatfrac`

`\partialderivative`

`\pderivative{x}`	$\pderivative{x} \to \frac{\partial}{\partial x}$	alternate name
`\pdv{x}`	$\pdv{x} \to \frac{\partial}{\partial x}$	shorthand name
`\pdv{f}{x}`	$\pdv{f}{x} \to \frac{\partial f}{\partial x}$	two arguments
`\pdv[n]{f}{x}`	$\pdv[n]{f}{x} \to \frac{\partial^n f}{\partial x^n}$	optional power
`\pdv{x}(\grande)`	$\pdv{x}(\grande) \to \frac{\partial}{\partial x}\left(\blacksquare\right)$	long-form
`\pdv{f}{x}{y}`	$\pdv{f}{x}{y} \to \frac{\partial^2 f}{\partial x \partial y}$	mixed partial
`\pdv*{f}{x}`	$\pdv*{f}{x} \to \partial f / \partial x$	inline form using `\flatfrac`

`\variation`

`\var{F[g(x)]}`	$\var{F[g(x)]} \to \delta F[g(x)]$	functional variation (works like `\dd`)
`\var(E-TS)`	$\var(E-TS) \to \delta(E-TS)$	long-form

`\functionalderivative`

`\fdv{g}`	$\fdv{g} \to \frac{\delta}{\delta g}$	functional derivative (works like `\dv`)
`\fdv{F}{g}`	$\fdv{F}{g} \to \frac{\delta F}{\delta g}$	
`\fdv{V}(E-TS)`	$\fdv{V}(E-TS) \to \frac{\delta}{\delta V}(E-TS)$	long-form
`\fdv*{F}{x}`	$\fdv*{F}{x} \to \delta F / \delta x$	inline form using `\flatfrac`

Matrices

The following matrix macros produce unformatted rows and columns of matrix elements for use as separate matrices as well as blocks within larger matrices. For example, the command `\identitymatrix{2}` which has also has the shortcut `\imat{2}` produces the elements of a 2×2 identity matrix $\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}$ without braces or grouping. This allows the command to also be used within another matrix, as in:

Example

```
\begin{equation}
  \begin{pmatrix}
    \imat{2} \\
    a & b
  \end{pmatrix}
\end{equation}
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ a & b \end{pmatrix} \tag{19}$$

To specify elements on the right of left sides of our `\imat{2}` sub-matrix we use the grouping command `\matrixquantity` or `\mqty` to effectively convert `\imat{2}` into a single matrix element of a larger matrix:

Example

```
\begin{equation}
  \begin{pmatrix}
    \mqty{\imat{2}} & \mqty{a\\b} \\
    \mqty{c & d}    & e
  \end{pmatrix}
\end{equation}
```

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ c & d & e \end{pmatrix} \tag{20}$$

The extra `\mqty` groups were required in this case in order to get the a and b elements to behave as a single element, since `\mqty{\imat{2}}` also acts like a single matrix element (the same can be said of the grouped c and d elements). Finally, the outermost `pmatrix` environment could have also been replaced with the `physics` macro `\mqty()`, allowing the above example to be written on one line:

Example

```
\begin{equation}
  \mqty(
    \mqty{\imat{2}} & \mqty{a\\b} \\
    \mqty{c & d}    & e
  )
\end{equation}
```

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ c & d & e \end{pmatrix} \tag{21}$$

The matrix commands are listed below:

`\matrixquantity`

`\mqty{a & b \\ c & d}` → $\begin{matrix} a & b \\ c & d \end{matrix}$ groups a set of matrix elements into a single object

`\mqty(a & b \\ c & d)` → $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ parentheses

`\mqty*(a & b \\ c & d)` → $\left(\begin{matrix} a & b \\ c & d \end{matrix}\right)$ alternate parentheses

`\mqty[a & b \\ c & d]` → $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ square brackets

`\mqty|a & b \\ c & d|` → $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ vertical bars

`\pmqty{}` ↔ `\mqty()`
`\Pmqty{}` ↔ `\mqty*()`
`\bmqty{}` ↔ `\mqty[]`
`\vmqty{}` ↔ `\mqty||`

alternative syntax; robust and more LaTeX-friendly

`\smallmatrixquantity` `\smqty{a & b \\ c & d}` → $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ the `smallmatrix` form of `\mqty`

`\smqty()` or `\spmqty{}` small version of `\mqty()`
`\smqty*()` or `\sPmqty{}` small version of `\mqty*()`
`\smqty[]` or `\sbmqty{}` small version of `\mqty[]`
`\smqty||` or `\svmqty{}` small version of `\mqty||`

`\matrixdeterminant` `\mdet{a & b \\ c & d}` → $\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ matrix determinant

`\smdet{a & b \\ c & d}` → $\left|\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right|$ small matrix determinant

`\identitymatrix` `\imat{n}` elements of $n \times n$ identity matrix

`\smqty(\imat{3})` → $\left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right)$ formatted with `\mqty` or `\smqty`

`\xmatrix` `\xmat{x}{n}{m}` — elements of $n \times m$ matrix filled with x

`\smqty(\xmat{1}{2}{3})` $\rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ — formatted with `\mqty` or `\smqty`

`\smqty(\xmat*{a}{3}{3})` $\rightarrow \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$ — star for element indices

`\smqty(\xmat*{a}{3}{1})` $\rightarrow \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ — as a vector with indices

`\smqty(\xmat*{a}{1}{3})` $\rightarrow \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}$

`\zeromatrix` `\zmat{n}{m}` — $n \times m$ matrix filled with zeros

`\smqty(\zmat{2}{2})` $\rightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ — equivalent to `\xmat{0}{n}{m}`

`\paulimatrix` `\pmat{n}` — n^{th} Pauli matrix

`\smqty(\pmat{0})` $\rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ — $n \in \{0, 1, 2, 3 \text{ or } x, y, z\}$

`\smqty(\pmat{1})` $\rightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

`\smqty(\pmat{2})` $\rightarrow \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$

`\smqty(\pmat{3})` $\rightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

`\diagonalmatrix` `\dmat{a,b,c,...}` — specify up to eight diagonal or block diagonal elements

`\mqty(\dmat{1,2,3})` $\rightarrow \begin{pmatrix} 1 & & \\ & 2 & \\ & & 3 \end{pmatrix}$

`\mqty(\dmat[0]{1,2})` $\rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ — optional argument to fill spaces

`\mqty(\dmat{1,2&3\\4&5})` $\rightarrow \begin{pmatrix} 1 & & \\ & 2 & 3 \\ & 4 & 5 \end{pmatrix}$ — enter matrix elements for each block as a single diagonal element

`\antidiagonalmatrix` `\admat{a,b,c,...}` — same as syntax as `\dmat`

`\mqty(\admat{1,2,3})` $\rightarrow \begin{pmatrix} & & 1 \\ & 2 & \\ 3 & & \end{pmatrix}$

The systeme Package

To use the systeme package, simply insert the command in the preamble of your document.

Command

```
\usepackage{systeme}
```

This package can make it really easy when typesetting linear systems by the command \systeme.

Example

```
\begin{equation}
  \systeme{
    2a-3b+4c=2,
    a+8b+5c=8,
    -a+2b+c=-5
  }
\end{equation}
```

$$\begin{cases} 2a-3b+4c=2 \\ a+8b+5c=8 \\ -a+2b+c=-5 \end{cases} \tag{22}$$

It also works for subscripts.

Example

```
\begin{equation}
  \systeme{
    4x_1-x_2=3,
    -x_1+5x_2=-1
  }
\end{equation}
```

$$\begin{cases} 4x_1 - x_2 = 3 \\ -x_1 + 5x_2 = -1 \end{cases} \tag{23}$$

It can also reorder the variables and numbers in the equations.

Example

```
\begin{equation}
  \systeme{
    3y+2x=0,
    x-z+9=0,
    2+3x+5-y-7+z=0
  }
\end{equation}
```

$$\begin{cases} 2x + 3y = 0 \\ x - z + 9 = 0 \\ 3x - y + z + 2 + 5 - 7 = 0 \end{cases} \tag{24}$$

Complicated coefficients can be handle correctly. Note that + and - should be replaced with \+ and \- in the coefficients.

Example

```
\begin{equation}
  \systeme{
    (2\+\sqrt{2})x-
    (1\-\sqrt{2})y=1,
    x+(1\+\sqrt{2})y=-1
  }
\end{equation}
```

$$\begin{cases} (2+\sqrt{2})x-(1-\sqrt{2})y=1 \\ x+(1+\sqrt{2})y=-1 \end{cases} \tag{25}$$

The documentation of the systeme package can be found in http://mirrors.ctan.org/macros/generic/systeme/systeme_fr.pdf, however it's in French, and the author is Manuel de l'utilisateur.

The gauss Package

To use the gauss package, simply insert the command in the preamble of your document.

Command

```
1  \usepackage{gauss}
```

This package provides LaTeX-macros for typesetting operations on a matrix. By an "operation on a matrix" we understand a row operation or a column operation. It is named gauss because Gauss Elimination is a widely used application of matrix operations.

The documentation of the systeme package can be found in http://mirrors.ctan.org/macros/latex/contrib/gauss/gauss-doc.pdf.

For example, if you are taking VV285 or working with other linear algebra stuffs in LaTeX, you may use the `gmatrix` environment provided by the `gauss` package.

Example

```
\begin{equation}
  \begin{array}{ccc|}
    4 & 2 & -2 \\
    -3 & 1 & 0 \\
    1 & 4 & 2
  \end{array}
  \begin{gmatrix}
    -2 \\ 6 \\ -9
    \rowops
    \swap{0}{2}
    \add[*(3)]{0}{1}
  \end{gmatrix}
\end{equation}
```

$$\left.\begin{array}{ccc|c} 4 & 2 & -2 & -2 \\ -3 & 1 & 0 & 6 \\ 1 & 4 & 2 & -9 \end{array}\right. \begin{array}{l} \leftarrow \\ \leftarrow \\ \leftarrow \end{array} \begin{array}{l} \\ *(3) \\ + \end{array} \tag{26}$$

Lecture IV

Graphs, Tables and Code

Include Graphs

Before all, you need the `graphics` or `graphicx` package, where `graphicx` is an extended and enhanced one. So you are recommended to insert the command in the preamble of your document.

Command

```
\usepackage{graphicx}
```

Then you can use the command `\includegraphics` to insert images of many formats, including `jpg`, `png` images and even other `pdf` files. `eps` images should be supported by most modern LaTeX distributions as well.

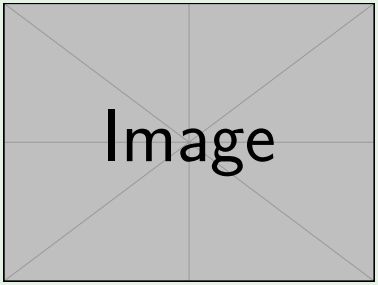
Command

```
\includegraphics[options]{filename}
```

There are some example images defined, you can insert them if the figure is not yet ready when writing LaTeX code. They are `example-image`, `example-image-golden`, `example-image-a`, `example-image-b` and etc.

Example

```
1  \includegraphics[width=0.4\textwidth]{example-image}
```



We usually use the `width` option to adjust the size of the image, according to a ratio of `\textwidth`, which means the maximum width of text here.

Options of Include Graphs

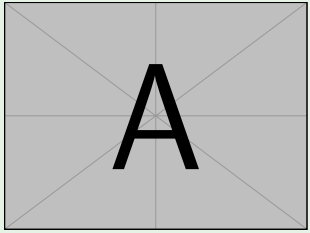
Here some useful `options` are listed:

- `height` - use any LaTeX measuring unit.
- `width` - use any LaTeX measuring unit.
- `scale` - scale the graph to this proportion
- `angle` - rotate the graph in anti-clockwise by this angle

LaTeX measuring unit can be `\textwidth`, `\linewidth`, `\textheight`, `\lineheight`, cm, pt, em, and etc..

Example

```
1  \includegraphics[width=4cm]%
2  {example-image-a}
```



The figure Environment

The figure environment provides a wrapper of image inserted by \includegraphics, which add caption and label (reference) to an image. They are especially useful in report and paper writing, here is a template of how to use the environment.

Command

```
\begin{figure}[position]
  \centering
  \includegraphics[options]{filename}
  \caption{caption}
  \label{fig:label}
\end{figure}
```

- filename - the filename or relative path of the graph you want to insert, usually placed in the same or child directory as the tex file
- position - we usually use !htbp or !H here, which will be introduced later in this chapter
- caption - the caption displayed above/under the graph
- label - used for references in a document (will be introduced later)

Labels and References

You can use `\ref` to have a reference of a figure by its label. The figures will be automatically numbered (like equations), and the reference is also a hyperlink.

Example

```
\begin{figure}[!htbp]
  \centering
  \includegraphics[
    width=0.8\textwidth,
    angle=90
  ]{example-image-b}
  \caption{Example Image B rotated by 90
  ↪ degree.}
  \label{fig:img-b}
\end{figure}
B was shown in Figure
\ref{fig:img-b}.
```

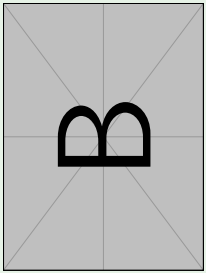


Figure 2: Example Image B rotated by 90 degree.

B was shown in Figure 2.

Floats and Positions

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes `figure` and `table` (will be introduced later) floats.

If you don't provide the `position` option, LaTeX will try to help you find a place to set the figure. However, the position is often not ideal, so you need to add some specifiers yourselves.

- `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, i.e., `\usepackage{float}`.

Include Multiple Graphs

A useful extension is the `subcaption` package, which provides a `subfigure` environment to add multiple subfigures in a figure.

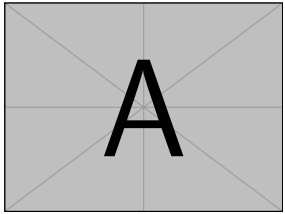
Note that there is also a package called `subfigure`, but is has been deprecated (not maintained), please do not use it. Another package called `subfig` provides the same commands as that of `subfigure` package. However, they can't be used together.

In simplicity, if there is some compatibility problem with your template after you tried the `subcaption` package, choose the `subfig` package.

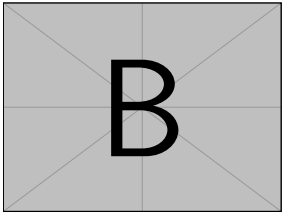
Here is an example with the `subcaption` package.

Example

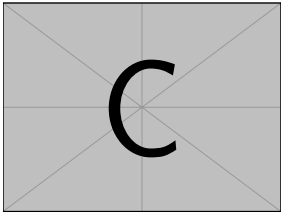
```
\begin{figure}
    \centering
    \begin{subfigure}{0.3\textwidth}
        \includegraphics[width=\textwidth]{example-image-a}
        \caption{Example Image A.}
        \label{fig:subcaption-a}
    \end{subfigure}
    ~
    \begin{subfigure}{0.3\textwidth}
        \includegraphics[width=\textwidth]{example-image-b}
        \caption{Example Image B.}
        \label{fig:subcaption-b}
    \end{subfigure}

    \begin{subfigure}{0.3\textwidth}
        \includegraphics[width=\textwidth]{example-image-c}
        \caption{Example Image C.}
        \label{fig:subcaption-c}
    \end{subfigure}
    \caption{Example Images}\label{fig:subcaption}
\end{figure}
```



(a) Example Image A.



(b) Example Image B.



(c) Example Image C.

Figure 3: Example Images

As shown in Figure 3, the figures can be arranged in columns and rows.

Between Figure 3a and Figure 3b, a `~` was added. You can add desired spacing between images, e. g. `~`, `\quad`, `\qquad`, `\hfill` (fill all rest horizontal spaces) and etc..

Between Figure 3b and Figure 3c, a newline was added. It will force the subfigure onto a new line.

The references of subfigures can be used by their `\label` as well. For example, above references are generated by these commands:

Example

```
\ref{fig:subcaption}
\ref{fig:subcaption-a}
\ref{fig:subcaption-b}
\ref{fig:subcaption-c}
```

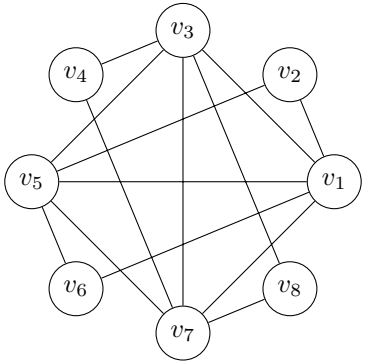
The tikz and pgf packages

The tikz and pgf packages can help you draw graphs in LaTeX for example:

Example

```
\begin{tikzpicture}[scale=2, bend angle=22.5]
\tikzstyle{every node}=[draw,shape=circle];
\foreach \i in {1,...,8}
{
\path (45*\i-45:1cm) node (v\i) {$v_\i$};
}
\draw
(v1) -- (v2) (v3) -- (v4) (v5) -- (v6) (v7) -- (v8)
(v1) -- (v3) (v3) -- (v5) (v5) -- (v7) (v7) -- (v1)
(v2) -- (v5) (v4) -- (v7) (v6) -- (v1) (v8) -- (v3)
(v1) -- (v5) (v3) -- (v7);
\end{tikzpicture}
```

This will generate a simple graph which consists of eight nodes:



There may be a lecture about `tikz` and `pgf` in the future. If you are now interested in it, please refer to the pgf manuel by `texdoc tikz` or `texdoc pgf`.

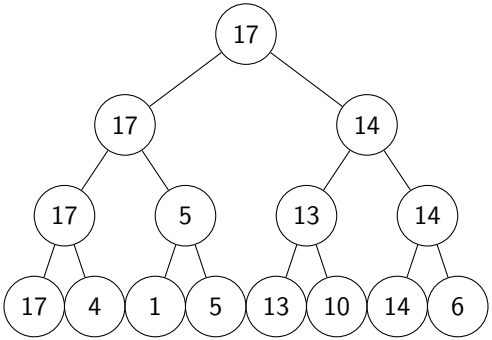
Another example:

Example

```
\begin{tikzpicture}[scale=0.8]
\tikzstyle{every node}=[draw,shape=circle,minimum size=0.8cm];
\node {17}[sibling distance=4cm]
child { node {17}[sibling distance=2cm]
    child {
        node {17}[sibling distance=1cm]
        child { node {17} }
        child { node {4} }
    }
    child {
        node {5}[sibling distance=1cm]
        child { node {1} }
        child { node {5} }
    }
}
child { node {14}[sibling distance=2cm]
    child {
        node {13}[sibling distance=1cm]
        child { node {13} }
        child { node {10} }
```

```
21        }
22        child {
23            node {14}[sibling distance=1cm]
24            child { node {14} }
25            child { node {6} }
26        }
27    };
28    \end{tikzpicture}
```

This will generate a binary tree:



The tabular Environment

Table is another common element in LaTeX, usually you will need the array package for enhanced functions of tables. You can insert the command in the preamble of your document.

Command

```
\usepackage{array}
```

Example

```
\begin{tabular}{|l|c|r|}
  \hline
  Title 1 & Title 2 & Title 3 \\
  \hline
  1 & 2 & 3 \\
  \hline
\end{tabular}
```

Title 1	Title 2	Title 3
1	2	3

The syntax is similar to the align environment in maths. & is used to split the columns are \\ is used to split the rows.

Column Format

Command

```
\begin{tabular}{format}
...
\end{tabular}
```

`format` can be set as follow:

- `|` - represents a vertical separate line between two columns
- `l` - align left in this column
- `c` - align center in this column
- `r` - align right in this column

Example

`|l|l|l|`

Title 1	Title 2	Title 3
1	2	3

`||c|cc||`

Title 1	Title 2	Title 3
1	2	3

With the help of the `array` package, more formats are available:

- `p{width}` - Equivalent to `\parbox[t]{width}`, vertically aligned bottom
- `b{width}` - Equivalent to `\parbox[b]{width}`, vertically aligned top
- `m{width}` - Equivalent to `\parbox{width}`, vertically aligned middle
- `>{decl.}` - Can be used after a letter option, inserts `decl` before the entry.
- `<{decl.}` - Can be used before a letter option, inserts `decl` after the entry.

`t` and `b` may be very confusing, but that's how they work in `\parbox`. With these new formats, the columns can be defined more flexibly.

Example

```
1  \begin{tabular}
2  {|p{1.2cm}|b{1.2cm}|m{1.2cm}|}
3    \hline
4    Aligned Bottom & Aligned Top &
5    Aligned Middle \\
6    \hline
7    1 & 2 & 3 \\
8    \hline
9  \end{tabular}
```

Aligned Bottom	Aligned Top	Aligned Middle
1	2	3

`t`, `b` and `m` only affect the vertical alignment. If you want to control the width and make the text horizontally centered as well, you can use `>{\centering}` to insert a `\centering` before the text in that column. You can also insert `>{$}` and `<{$}` to generate a column in math mode.

Example

```
\begin{tabular}{|>{\centering}m{2cm}|>{$}b{2cm}<{$}|}
  \hline
  Row of Text &
  \text{Row of Maths} \\
  \hline
  First & x \\
  Second & x^2 \\
  \hline
\end{tabular}
```

Row of Text	Row of Maths
First	x
Second	x^2

If a column type will be used many times, and also very long, you can define a new column type by yourselves. You can use

Command

```
\newcolumntype{new type}{>{some declarations}{old type}<{some more declarations}}
```

If you want to repeat a format for multiple times, you can use `*{num}{format}`. Here's an example of the usage of `\newcolumntype` with multiple columns form.

Example

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
\newcolumntype{R}{>{$}r<{$}}

\begin{tabular}{|L| *{2}{C|} R|}
  \hline
  \text{First} & \text{Second} &
  \text{Second} & \text{Third} \\
  \hline
  x & x^2 & x^2 & x^3 \\
  \hline
  y & y^2 & y^2 & y^3 \\
  \hline
\end{tabular}
```

First	Second	Second	Third
x	x^2	x^2	x^3
y	y^2	y^2	y^3

Horizontal Lines

We usually need horizontal lines in tables. As shown in the examples above, you can add a `\hline` at the beginning of a row.

If you only want to draw a partial line, use `\cline[start-end]`.

Example

```
\begin{tabular}{c|l|c|r}
  \hline\hline
  & Title 1 & Title 2 & Title 3 \\
  \cline{2-4}
  Table & 1 & 2 & 3 \\
  \cline{2-4}
  & 4 & 5 & 6 \\
  \hline\hline
\end{tabular}
```

	Title 1	Title 2	Title 3
Table	1	2	3
	4	5	6

Here we draw a table with a multirow, but it only works with multirows of odd row number. A more convenient method of drawing multirows will be introduced.

Combine Rows and Columns

There are two commands being used to combine rows and columns

Command

`\multicolumn{ncols}{format}{text}`

- `ncols` - the number of columns to be merged
- `format` - the format of the merged column, excluding the left `|` (eg. `c|`)
- `text` - the text in the merged column

`\multirow{nrows}{width}[fixup]{text}`

- `nrows` - the number of rows to be merged
- `width` - the width of the merged rows (use `*` for auto)
- `fixup` - the vertical position of the text (optional, default in the center)
- `text` - the text in the merged row

To use the `\multirow` command, you need to insert the package `multirow` in the preamble of your document.

Example

```
\centering
\begin{tabular}{|c|c|c|c|c|}
  \hline
  \multirow{4}{*}{Table} & Title 1 & Title 2 & Title 3 & Title 4 \\
  \cline{2-5}
  & \multicolumn{2}{c|}{Text 1} &
  \multicolumn{2}{c|}{\multirow{3}{*}{Text 3}} \\
  \cline{2-3}
  & \multicolumn{2}{c|}{Text 2} & \multicolumn{2}{c|}{} \\
  \cline{2-3}
  & Text 4 & Text 5 & \multicolumn{2}{c|}{} \\
  \hline
\end{tabular}
```

<table>
<tr><td rowspan="4">Table</td><td>Title 1</td><td>Title 2</td><td>Title 3</td><td>Title 4</td></tr>
<tr><td colspan="2">Text 1</td><td colspan="2" rowspan="3">Text 3</td></tr>
<tr><td colspan="2">Text 2</td></tr>
<tr><td>Text 4</td><td>Text 5</td></tr>
</table>

Just leave blank in the rest rows of `\multirow`.

Table Generators

With `\multirow` and `\multicolumn`, we can almost draw tables of any style, but this coding process can never be as easy as the graphic one, like making tables in Word or Excel. Is there any ways to convert graphic tables into LaTeX codes directly?

- Use LaTeX Table Generator: http://www.tablesgenerator.com/
- LaTeX Complex Table Editor: https://www.latex-tables.com/
- Excel2latex: https://ctan.org/tex-archive/support/excel2latex/

The table Environment

The table environment is used to arrange the place of a tabular, similar to the figure environment. Here is a template of how to use the environment.

Command

```
\begin{table}[position]
  \centering
  \begin{tabular}{format}
    ...
  \end{tabular}
  \caption{caption}
  \label{table:label}
\end{table}
```

The position, caption, label are same as those in the figure environment.

Recall the Positions

We usually want to place the graphs or tables just below or above the content where we mention them, but even when we type `[h]` in position, you can not ensure that it will appear at the ideal position, and there are several methods to make up for this. You can try them one by one:

1. Change `[h]` to `[!h]`
2. Change `[!h]` to `[!H]`
3. Use `\newpage` to move the following content to the next page

Usually you don't need to pay too much attention about where the figures and tables are exactly are because you can use `\ref` to reference them. And the numbering of figures and tables will strictly follow the order of their code.

figure and table in Two-column Documents

If you are writing a document using two columns (i.e. you started your document with something like `\documentclass[twocolumn]{article}`), you might have noticed that you can't use floating elements that are wider than the width of a column (using a LaTeX notation, wider than `0.5\textwidth`), otherwise you will see the figure or table overlapping with text.

If you really have to use such wide elements, the only solution is to use the "starred" variants of the floating environments:

Command

```
\begin{figure*}[position]
  ...
\end{figure*}
```

```
\begin{table*}[position]
  ...
\end{table*}
```

Those "starred" versions work like the standard ones, but they will be as wide as the page, so you will get no overlapping.

The array Environment

When you use tabular in maths environment, the text format in the tabular won't be italic. However, there is a replacement of tabular, which is the array environment.

Command

```
1  \begin{array}{format}
2    ...
3  \end{array}
```

The options and usages of these two environment are exactly the same.

Though the environment is not provided by the array package (it's built-in one), you are also recommended to use this package for enhancements.

The minted Package

All of the code in this lecture are highlighted by the minted package. To use it, simply insert the command in the preamble of your document.

Command

```
\usepackage{minted}
```

This is a very special package, it depends a program out of LaTeX called pygmentize, which is a code highlighting package written in Python.

You can install the package through pip (assuming you have Python 2 or 3 and pip installed) in your terminal:

Command

```
pip install Pygments
```

And then you can examine in your terminal whether pygmentize is your PATH by directly running it. You also need to add an option -shell-escape to your LaTeX compiler because LaTeX need this permission to run other programs on shell.