

L^AT_EX 学习手册

1 L^AT_EX 基础

1.1 L^AT_EX 文档结构

1. 文档结构：以下是一份最为简单的 L^AT_EX 文档：

```
\documentclass{article}
\begin{document}
Hello, \LaTeX
\end{document}
```

Hello, L^AT_EX

2. 命令：在 L^AT_EX 中，命令也称控制序列 (control sequence)，以一个反斜杠加上命令名构成。开头的命令 `\documentclass` 指定了使用的文档类，花括号 `{ }` 内的内容是命令的参数，`article` 表示文章格式的文档类，除此之外可用的参数还有表示报告的文档类 `report` 和表示书籍的文档类 `book` 等。

如果命令的参数只有一个字符，则花括号可省略，但需要用空格区分命令与参数。

有些命令还存在可选参数，可选参数通常会在花括号前用方括号 `[]` 表示，例如：

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

多数命令只在原地产生效果，但有些命令则会影响到作用域内后面的所有内容，这种命令又称为声明 (declaration)。

`\begin` 和 `\end` 一对命令定义了一个环境，环境表示命令或内容的作用范围。环境如果有备选或额外参数，只需在 `\begin` 中表示。`document` 环境当中的内容是文档正文，在此环境外书写的内容可能不会出现在文档中。

语句 `\begin{document}` 之前的内容称为导言区，导言区可以留空，也可以编写文档所需要的信息与工具。

`\LaTeX` 命令用于输出一个特殊的符号 L^AT_EX，像这样表示符号的命令还有很多。

3. 注释：以百分号 `%` 开头的部分是行注释。

注释有时放在行尾，用于取消换行产生的一个多余的空格。

如果要单独表示百分号，需要使用反斜杠转义 `\%`

4. 单位：L^AT_EX 中常用的衡量长度的单位有：

- pt : 磅 (point)
- pc : 四号字 (pica) $1\text{ pc} = 12\text{ pt}$
- in : 英寸 (inch) $1\text{ in} = 72.27\text{ pt}$
- bp : 大点 (bigpoint) $1\text{ bp} = \frac{1}{72}\text{ in}$
- cm : 厘米 (centimeter) $1\text{ cm} = \frac{1}{2.54}\text{ in}$
- mm : 毫米 (millimeter)
- sp : T_EX 的基本长度单位 scaled point , $1\text{ sp} = \frac{1}{65536}\text{ pt}$
- em : 当前字号下大写字母 M 的宽度
- ex : 当前字号下小写字母 x 的高度

5. 宏包：宏包的作用是扩展或调整 L^AT_EX 的排版功能。一个宏包往往能提供更多的命令、环境，或为内置的命令/环境添加更多功能。

在导言区使用 `\usepackage{}` 即可引入宏包，然后便可以使用宏包提供的功能。部分宏包在引入时可以通过命令的可选参数调节需要引入的功能。

例如，引入 `\ctex` 宏包可以在文档中排版中文字符，引入 `\amssymb` 宏包可以使用命令 `\bigstar` 表示一个填充五角星符号 ★

1.2 特殊符号

1. 空格：在 \LaTeX 中，字符间的空白会自动调整。

不带参数的命令后面的空格，要用空的花括号对加上空格来表示，否则空格会被忽略，例如：

<code>\LaTeX without \{\}</code>	$\text{\LaTeX}without \{ \}$
<code>\LaTeX{} with \{\}</code>	$\text{\LaTeX} with \{ \}$

2. 引号：英文单引号分左右引号。左单引号用重音符 ``` 表示，右单引号用普通引号 `'` 表示，左双引号用连续两个重音符 ```` 表示，右双引号用连续两个单引号 `''` 表示。

英文下的引号嵌套需要借助 `\thinspace` 命令分隔，例如：

<code>``\thinspace`single' quotes''</code>	“‘single’ quotes”
--	-------------------

中文引号可以直接输入。

3. 短横：英文短横有 3 种：

- 连字符：用一个短横 `-` 表示，如 clear-cut ；
- 数字起止符：用两个短横 `--` 表示，如 page 1-2 ；
- 破折号：用三个短横 `---` 表示，如 Look—It’s a dash 。

中文破折号可以直接输入。

4. 省略号：英文省略号用 `\ldots` 符号表示，效果为 ...。中文省略号可以直接输入。
5. 保留字符：以下字符在 \LaTeX 中具有特殊含义，不能直接作为文档中的一个字符：

<code># \$ % ^ & _ { } \</code>

除反斜杠外，其余字符均能用反斜杠的形式转义输出：

<code>\# \\$ \% \^{} \& _ \{ \}</code>	<code># \$ % ^ & _ { }</code>
---	-----------------------------------

反斜杠 `\` 可以使用以下方式得到：

<code>\textbackslash \$\backslash\$</code>
--

1.3 文字样式

1. 粗体与斜体：广义的斜体命令是 `\textit{}`，粗体命令是 `\textbf{}`。`\emph{}` 命令用于强调文本，对西文字母而言就是变为斜体。例如：

<code>\emph{text} and \textbf{text}</code>	<i>text</i> and text
--	-----------------------------

2. 字体样式：可以通过以下声明修改字体的字族、字系、字形效果，这三种类型声明相互独立，可以组合使用：

字族	<code>\rmfamily</code>	- 罗马字族 Roman
	<code>\sffamily</code>	- 无衬线字族 Sans Serif
	<code>\ttfamily</code>	- 罗马字族 Typewriter
字系	<code>\bfseries</code>	- 粗体 Bold Font
	<code>\mdseries</code>	- 中粗体 Middle
字形	<code>\upshape</code>	- 竖直 Upshape
	<code>\slshape</code>	- 斜体 <i>Slant</i>
	<code>\itshape</code>	- 意大利体 <i>Italic</i>
	<code>\scshape</code>	- 小号大写体 SMALLCAP

但如果不存在这样的字体设计，某些效果可能不会生效。

这些声明会影响之后的所有文本，如果只是改变局部字体样式，可以使用 `\text{...}` 这类命令，例如 `\textbf{}`。

3. 字号：在 `\documentclass` 的可选参数可以指定正文字号大小，参见对可选参数的第一次介绍。可以通过以下声明相对地改变字号大小：

```

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

```

4. 基本下划线：原生的 下划线命令 是 `\underline`，上划线命令 是 `\overline`，它们均需要在公式环境 (参见 2.1 节) 中使用。

5. 更好的下划线：`\usepackage{ulem}` 宏包提供了更好的下划线命令，包括：

命令	效果	命令	效果
<code>\uline{}</code>	<u>下划线</u>	<code>\uuline{}</code>	<u>双下划线</u>
<code>\dashuline{}</code>	<u>虚线下划线</u>	<code>\dotuline{}</code>	<u>点下划线</u>
<code>\uwave{}</code>	<u>波浪线</u>	<code>\sout{}</code>	删除线
<code>\xout{}</code>	<u>斜删除线</u>		

这些命令可以直接使用。

`ulem` 宏包修改了 `\emph` 命令的效果，会给强调文本加下划线，可能不是想要的效果，可以通过宏包的 `[normalen]` 选项取消该效果。

1.4 段落处理

1. 文本对齐：有左中右三种对齐方式，如下表：

对齐方式	声明形式	环境形式	参数形式
居中对齐	<code>\centering</code>	<code>center</code> 环境	<code>\centerline{}</code>
左对齐	<code>\raggedright</code>	<code>flushleft</code> 环境	<code>\leftline{}</code>
右对齐	<code>\raggedleft</code>	<code>flushright</code> 环境	<code>\rightline{}</code>

声明形式的对齐会影响接下来所有文本的对齐方式，环境形式只影响环境内的对齐方式，参数形式只影响参数内文本的对齐方式。

- 2. 换行： \LaTeX 中的多个空格会被视为一个，多个换行符也会被视为一个。使用两个回车可用于在正文中换行。使用两个反斜杠 \LaTeX 用于强制换行。
- 3. 段落与缩进：使用 \LaTeX 可以生成一个带缩进的新段。
- 4. 换页：使用 \LaTeX 开始新的一页。

2 数学公式

2.1 引入公式

- 1. 行内公式：行内公式将嵌入到文本行中，公式垂直距离不会过高。行内公式有 3 种引入方式：

- $\$ \dots \$$
- $\text{\LaTeX}(\dots \text{\LaTeX})$
- $\text{\LaTeX}\begin{math} \dots \end{math}$

例如：

$\$ a + b = c \$$	$a + b = c$
-------------------	-------------

- 2. 行间公式：行间公式将单独出现并居中，垂直距离将适应公式内容。行间公式有 3 种引入方式：

- $\$ \$ \dots \$ \$$
- $\text{\LaTeX}[\dots \text{\LaTeX}]$
- $\text{\LaTeX}\begin{displaymath} \dots \end{displaymath}$

例如：

$\text{\LaTeX}[\sum_{i=1}^n x_i \text{\LaTeX}]$	$\sum_{i=1}^n x_i$
---	--------------------

如果用行内公式表达它，受行高限制呈现的效果稍有不同，接下来给出示例：

- 3. 公式尺寸：公式在不同位置会呈现不同尺寸，如果要想公式强制排版为特定的尺寸，可以使用以下几个声明：

命令	尺寸	示例
$\text{\LaTeX}\displaystyle$	行间公式尺寸	$\sum_{i=1}^n x_i$
$\text{\LaTeX}\textstyle$	行内公式尺寸	$\sum_{i=1}^n x_i$
$\text{\LaTeX}\scriptstyle$	上下标公式尺寸	$\sum_{i=1}^n x_i$
$\text{\LaTeX}\scriptscriptstyle$	次上下标尺寸	$\sum_{i=1}^n x_i$

- 4. 下标与上标：下标使用 \LaTeX 符号，上标使用 \LaTeX 符号，例如 $\text{\LaTeX} a_n$ 或 $\text{\LaTeX} e^x$
如果上下标内不止一个字符，这些字符需要用花括号 \LaTeX 定界，例如 $\text{\LaTeX} x^{10}$ 。
- 5. 数学符号：在数学公式内的字母将会变为斜体。一些基本函数如 \LaTeX 需要使用命令表示，从而用正体表示字母：

$\text{function} \text{\LaTeX} \sin x$	$\text{function} \sin x$
--	--------------------------

2.2 基本公式排版

1. 基本符号：加号 `+` + 减号 `-` 和等号 `=` = 直接使用对应字符创建，其余四则运算符号需要使用命令创建：乘号 `\times` × 除号 `\div` ÷ 不等号 `\neq` ≠
2. 分式：使用命令 `\frac{}{}`，两个参数分别表示分子和分母，例如：

`\[\frac{e^x}{\frac{a}{b}} \]`

$$\frac{e^x}{\frac{a}{b}}$$

同样，如果分子或分母内不止一个字符，需要用花括号定界。

3. 导数：直接使用单引号 `'` 来表示：

`$ f'(x) = a^x \ln x $`

$$f'(x) = a^x \ln x$$

4. 三种带上下限的特殊符号：求和 `\sum` Σ、求积 `\prod` Π、积分 `\int` ∫，它们均使用下标 `_` 符号和上标 `^` 符号来表示其上下限，例如：

`\sum_{i=1}^N a_i`

$$\sum_{i=1}^N a_i$$

5. 根号：使用 `\sqrt` 命令，配合可选参数可以得到不同次数的根式：

`\sqrt{9} \sqrt[3]{x}`

$$\sqrt{9} \sqrt[3]{x}$$

6. 极限：用 `\lim` 表示极限运算符，使用下标的形式表示底下的趋近关系，其中使用 `\to` 命令表示趋近的箭头，例如：

`\[\lim_{x \to \infty} f(x) \]`

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

2.3 公式的细节

1. 空格：源文件中在数学公式中的空格会被忽略。可以通过以下几种命令向公式中添加空格：

命令	空格大小	命令	空格大小
<code>\,</code>	3/18 空格	<code>\:</code>	4/18 空格
<code>\;</code>	5/18 空格	<code>\!</code>	−3/18 空格
<code>\quad</code>	1 空格	<code>\qquad</code>	2 空格
<code>_</code>	9/18 空格	(反斜杠加空格)	

负数尺寸的空格会拉进两个字符的距离。

2. 数学运算符与数学关系符：数学运算符和数学关系符区别在于两侧的间距。例如 `$a+b$` 表现为 $a + b$ ，但 `$+b$` 表现为 $+b$ 。前者的符号 + 是数学运算符，而后的符号 + 是正号符号。数学运算符相比普通符号，左右两侧间距更大。数学关系符类似，且两侧间距比运算符略大，如 `$a<b$` 中的关系符表现为 $a < b$ 。

数学模式中花括号 `{}` 内的公式将独立考虑符号关系，可以与两侧符号隔离，使之不成为运算符或关系符，从而取消间距，如 `$a{+}b$` 表现为 $a+b$ 。

普通符号两侧没有预留间距，命令 `\mathbin{}` 与 `\mathrel{}` 则能分别把参数转换为二元运算符和二元关系符，并正确设置两侧的空距，在对齐时用处较大。

3. 公式与标点符号：公式中常用的标点符号有 `,;`，它们与普通符号的区别为只与右侧符号有间距。普通的冒号 `:` 得到的是数学关系符，而命令 `\colon` 得到的才是标点符号中的冒号，且间距左小右大，参见 `$a:b$` $a:b$ 与 `$a\colon b$` $a:b$ 的区别。

命令 `\mathpunct{}` 能把参数作为标点符号处理，并正确设置左无右有的间距。

4. 操作符与上下标：数学操作符的上下标作为行间公式时，将会显示在正下方，而不是右上下方，例如 `\min_i` 会显示为 \min_i ，`\mathop{}` 命令可以将参数转换为操作符，例如 `x_n^2` 与 `\mathop{x}_n^2` 的区别
`\limits` 命令用在行内公式中，跟随在任意数学操作符后，可以将它的上下标显示在数学操作符的正上下方，并且保持行内公式的紧凑性，例如：

<code>\$\int\limits_a^b f(x)\mathrm{d}x\$</code>	$\int_a^b f(x)\mathrm{d}x$
--	----------------------------

5. 堆叠上下标：`\substack{}` 命令需要 `amsmath` 宏包支持，用于将多行符号堆叠为一个上下标，参数中可以使用两个下划线 `\\` 换行，例如：

<code>\[</code> <code>\lim_{\substack{x\to x_0 \\ y\to y_0}} f(x,y)</code> <code>\]</code>	$\lim_{\substack{x\rightarrow x_0 \\ y\rightarrow y_0}} f(x,y)$
--	---

6. 定界符：以下展示了一些常用的定界符：

命令	符号	命令	符号	命令	符号	命令	符号
	())				
<code>\lbrace or {</code>	{	<code>\rbrace or }</code>	}	<code>\lbrack or [</code>	[<code>\rbrack or]</code>]
<code>\lfloor</code>	⌊	<code>\rfloor</code>	⌋	<code>\lceil</code>	⌈	<code>\rceil</code>	⌋
<code>\langle</code>	⟨	<code>\rangle</code>	⟩	<code>\vert or </code>		<code>\Vert or \ </code>	

7. 定界符高度：使用 `\big` 等一系列命令及其与 `\r` 和 `\l` 的组合可以产生具有不同大小的括号：

<code>\$\$ (\big(\Big(</code> <code>\frac{ax+b}{\ln x}</code> <code>\bigg) \Bigg) \Biggr) \$\$</code>	$((\left(\frac{ax+b}{\ln x}\right)))$
---	---------------------------------------

使用 `\left`、`\right` 以及 `\middle` 能使定界符自适应公式的高度。`\left` 和 `\right` 必须成对出现以限定范围。单个点号表示的 `\left.` 和 `\right.` 仅用于配对以限定范围，不输出任何符号。

大于号 `>` 和小于号 `<` 也可以组合这些命令得到不同大小的尖括号。

8. 分式的各种表现：基本的 `\frac` 会在不同位置表现出不同的尺寸，`\tfrac` 和 `\dfrac` 分别创建 text 和 display 样式的分式，例如 `\dfrac{1}{1+x}` 在行内也显示为 $\frac{1}{1+x}$ ，这会撑起行高。

9. 连分式：使用 `\cfrac` 创建，以下是分别使用 `\frac`、`\dfrac` 和 `\cfrac` 创建的嵌套分式效果：

$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3}}}$	$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3}}}$	$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3}}}$
---	---	---

2.4 公式字体与字符

1. 粗体：数学环境中的粗体使用 `\boldsymbol{}` 命令，例如：

`\boldsymbol{y} = k \boldsymbol{x}`

$y = kx$

2. 正体：数学公式内的字体默认为斜体，使用命令 `\mathrm{}` 可以在公式内创建正体，例如 `\mathrm{argmin}` 效果为 argmin

3. 原生数学字体：下表列出了原生的数学字体：

命令	效果
<code>\mathrm{ABCDabcd1234}</code>	$\mathrm{ABCDabcd1234}$
<code>\mathit{ABCDabcd1234}</code>	$\mathit{ABCDabcd1234}$
<code>\mathnormal{ABCDabcd1234}</code>	$\mathnormal{ABCDabcd1234}$
<code>\mathcal{ABCDabcd1234}</code>	$\mathcal{ABCD}\mathrm{abcd1234}$

4. 其它数学字体：其它一些宏包支持的数学字体对 `ABCDabcd1234` 的排版效果为：

命令	效果	所需宏包
<code>\mathbb{ABCDabcd1234}</code>	$\mathbb{ABCD}\mathbb{abcd1234}$	<code>amssymb</code>
<code>\mathfrak{ABCDabcd1234}</code>	$\mathfrak{ABCD}\mathfrak{abcd1234}$	<code>amssymb</code>
<code>\mathscr{ABCDabcd1234}</code>	$\mathscr{ABCD}\mathscr{abcd1234}$	<code>mathrsfs</code>

5. 希腊字母：使用对应字母名表示，例如 `\alpha` α ，首字母大写表示大写希腊字母，例如 `\Omega` Ω 。有些希腊字母前面加上 `var` 表示花写，例如 `\varphi` φ 相比于 `\phi` ϕ 。

6. 正体小写希腊字母：宏包 `txfonts` 以 `up` 结尾的命令可以得到正体的小写希腊字母，如 `\thetaup` θ 相较于斜体的 `\theta` θ 。

宏包 `txfonts` 可能会改变其它数学符号甚至正文的字体样式，可以替换为宏包 `upgreek`，它以 `up` 开头的命令也可以得到正体小写希腊字母，如 `\uppi` π ，且不会有副作用。宏包 `upgreek` 有三个互斥的选项：`Euler`、`Symbol` 和 `Symbolsmallscale`，分别加载不同的字体或字号。

3 复合公式

3.1 矩阵

1. 基本矩阵：使用 `array` 环境可以得到表格一样的横竖对齐，可以用于对齐公式或排版矩阵。这些具有对齐的环境一般都使用 `\` 切换到下一行，同一行内用 `&` 切换到下一列：

`\[\begin{array}{ccc} x_{11} & ax + b & x_{13} \\ x_{21} & e^x & x_{23} \end{array} \]`

$$\begin{array}{ccc} x_{11} & ax + b & x_{13} \\ x_{21} & e^x & x_{23} \end{array}$$

额外的参数指明了每一列的水平对齐方式都是居中对齐 (center)，还可以使用 `l` 指定左对齐以及使用 `r` 指定右对齐。

2. 通用矩阵：宏包 `\usepackage{amsmath}` 提供了通用的 `\matrix` 矩阵环境，无需手动指明对齐方式，其余用法一致：

<pre>\[\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \]</pre>	$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$
---	--

3. 矩阵的边界符：可以用左右高度自适应的定界符为矩阵加上边界符。宏包 `\usepackage{amsmath}` 以 `\pBbV` 开头的各种 `\matrix` 环境可以为矩阵两侧加上各种边界符，以下是按顺序展示的各个边界符效果：

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \left\{ \begin{matrix} a & b \\ c & d \end{matrix} \right\} \quad \left| \begin{matrix} a & b \\ c & d \end{matrix} \right| \quad \left\| \begin{matrix} a & b \\ c & d \end{matrix} \right\|$$

4. 行内小矩阵：宏包 `\usepackage{amsmath}` 提供了 `\smallmatrix` 环境，可以得到行内公式的小矩阵，加上自适应的左右括号后为 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ 效果。
5. 矩阵的边界符与对齐：宏包 `\usepackage{mathtools}` 提供了各种带星的 `\matrix*` 环境，在拥有边界符的同时，可以通过可选参数手动指定列对齐：

<pre>\[\begin{pmatrix*}[r] 150 & -450 \\ 10 & 15 \end{pmatrix*} \]</pre>	$\begin{pmatrix} 150 & -450 \\ 10 & 15 \end{pmatrix}$
---	---

3.2 并列公式

1. 分段函数与公式并列：使用 `\cases` 环境，会自动生成一个比 `\left` 更紧凑的花括号，例如：

<pre>\[y=\begin{cases} x & (x \leq 1) \\ 2x-1 & (x > 1) \end{cases} \]</pre>	$y = \begin{cases} x & (x \leq 1) \\ 2x - 1 & (x > 1) \end{cases}$
--	--

`\cases` 环境行列可像矩阵一样对齐，但最多只允许包含两列。

2. 并列公式中的公式样式：`\cases` 环境下的公式默认为 text 样式，使用 `\usepackage{mathtools}` 宏包的 `\dcases` 环境可以得到 display 样式的内容。以下是两者的比较：

$$\begin{cases} \int_0^x \frac{dx}{x+1} & x > 0 \\ x+1 & x \leq 0 \end{cases} \quad \begin{cases} \int_0^x \frac{dx}{x+1} & x > 0 \\ x+1 & x \leq 0 \end{cases}$$

3. 左并列与右括号：使用 `\usepackage{mathtools}` 宏包的 `\rcases` 环境可以得到右花括号环境，效果为：

<pre>\[\begin{rcases} a > 0 \\ x_1=2, x_2=5 \end{rcases} y=x^2-7x+10 \]</pre>	$\left. \begin{matrix} a > 0 \\ x_1 = 2, x_2 = 5 \end{matrix} \right\} y = x^2 - 7x + 10$
---	---

4. 并列公式和并列文本：使用带星号的 `\dcases*` 环境可以使并列环境的第二列条件不是公式而是文本，例如：


```
\[a_n =\begin{dcases*}
2^n & n \text{ is odd}\\
n^2 & n \text{ is even}
\end{dcases*} \]
```

$$a_n = \begin{cases} 2^n & n \text{ is odd} \\ n^2 & n \text{ is even} \end{cases}$$

3.3 公式与编号

- 带编号的公式：普通的行间公式不带编号，使用 `equation` 环境可以创建带编号的公式：

```
\begin{equation}
a^2 + b^2 = c^2
\end{equation}
```

$$a^2 + b^2 = c^2 \quad (1)$$

`equation` 环境自带公式环境，可以直接输入公式。

- 连续编号的公式：`amsmath` 宏包的 `gather` 环境可以给多行公式编号，环境内使用两个连续的反斜杠 `\` 换行，例如：

```
\begin{gather}
(a+b)^2=a^2+2ab+b^2\\
(a-b)^2=a^2-2ab+b^2
\end{gather}
```

$$(a+b)^2 = a^2 + 2ab + b^2 \quad (2)$$

$$(a-b)^2 = a^2 - 2ab + b^2 \quad (3)$$

- 多行的行间公式：普通的行间公式无法这样换行，带星号的 `gather*` 环境可以创建多行的行间公式，且均不编号。
- 公式与子编号：`amsmath` 宏包使用 `subequations` 环境可以使环境内的公式编号变为子编号，例如：

```
\begin{subequations} \begin{gather}
\sin^2 x = \frac{1 - \cos 2x}{2}\\
\cos^2 x = \frac{1 + \cos 2x}{2}
\end{gather} \end{subequations}
```

$$\sin^2 x = \frac{1 - \cos 2x}{2} \quad (4a)$$

$$\cos^2 x = \frac{1 + \cos 2x}{2} \quad (4b)$$

- 自定义编号行为：在行尾使用命令 `\notag` 可以取消该行的编号；在行尾使用命令 `\tag{}` 可以手动指定公式的编号，参数内为编号内容，且该编号会自动添加圆括号；带星号的命令 `\tag*{}` 同样可以手动指定编号，且不会自动添加圆括号。例如：

```
\begin{gather}
a+b \\
b+c \notag \\
c+d \tag{ii} \\
d+e \tag*{$\star$} \\
\end{gather}
```

$$a + b \quad (5)$$

$$b + c$$

$$c + d \quad (ii)$$

$$d + e \quad \star$$

3.4 对齐公式

- 公式对齐：使用 `amsmath` 宏包的 `align` 环境，通过 `&` 符号确定对齐位置：

```
\begin{align} a(b+c) \\ &= ab + ac \\ &= ac + ab \\ &= a(c+b) \end{align}
```

$$\begin{aligned} a(b+c) &= ab + ac & (6) \\ &= ac + ab & (7) \\ &= a(c+b) & (8) \end{aligned}$$

`align` 环境实质是奇数列居右、偶数列居左的表格，因此不用像 `array` 环境需要给出列的数目和对齐参数，并且可以通过空列改变列的左右对齐方式。

`align` 环境会给每一行公式编号，带星的 `align*` 环境会取消所有编号。

2. 对齐与空白：`amsmath` 宏包的 `align` 环境列间会产生较大的空白，换用 `alignat` 环境可以得到紧凑的对齐。该环境需要指定一个数字作为参数，代表右左对齐的对数，计算方法为 \geq 每行最大的 `&` 数量加 1 后再除以 2。

使用带星号的 `alignat*` 环境可以不带编号。

例如，以下对齐：

```
\begin{alignat*}{3} x+2&&y=-5 \\ -&&y=7 \\ \end{alignat*}
```

$$\begin{aligned} x+2y &= -5 \\ -y &= 7 \end{aligned}$$

换用 `align` 环境会产生大量的空白，效果一言难尽，但可以从中看出 `alignat` 环境的参数数为 3：

$$\begin{array}{ccc} x+2 & y = - & 5 \\ - & y = & 7 \end{array}$$

3. 公式块环境：`gather`、`align` 和 `alignat` 环境必定占据一整行，公式块环境 `gathered`、`aligned` 和 `alignedat` 环境只占公式实际宽度，因此同一行中可以编写其它内容。但是公式块环境需要置于其它数学环境中，且不带任何编号。

4. 对齐与编号：将公式块环境嵌套在 `equation` 环境中可以使多行公式只有一个居中的编号，例如：

```
\begin{equation} \begin{aligned} e^x &= 1 + x + \frac{1}{2!}x^2 + \cdots \\ &\quad + \frac{1}{n!}x^n + \cdots \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \end{aligned} \end{equation}
```

$$\begin{aligned} e^x &= 1 + x + \frac{1}{2!}x^2 + \cdots + \frac{1}{n!}x^n + \cdots \\ &= \sum_{n=0}^{\infty} \frac{x^n}{n!} \end{aligned} \quad (9)$$

5. 左中右对齐：`amsmath` 宏包使用 `multline` 环境得到第一行左对齐、中间的行居中对齐、最后一行右对齐的公式环境：

```
\begin{multline} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \\ a_4 + b_4 \\ \end{multline}
```

$$\begin{aligned} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \\ a_4 + b_4 \end{aligned} \quad (10)$$

使用带星号的 `multline*` 环境可以去除最后一行产生的编号。

4 进阶内容

4.1 调节空白距离

1. 定长水平空白：以下几个水平空白在公式环境外也能产生相同的空白效果：

```
\, \: \; \! \quad \qquad
```

这些命令置于段首和段尾无效。

2. 自定义水平空白：`\hspace{}` 用于在两个字符间产生高度为零的水平空白，参数为产生空白的水平距离。水平距离可以为负值，表示拉近间距。

该命令置于段首有效，置于段尾无效。如果命令位于换行处（左右字符输出后位于行尾和行首）将失效。

带星号的 `\hspace*{}` 作用基本相同，但它在换行处也可以工作，一定能在其左右字符之间生成空白。

3. 虚位水平空白：`\hphantom{}` 同样在两个字符之间产生空白，但空白宽度为参数中内容的水平宽度。换句话说它为参数的内容“预留水平空白”。该命令置于段首和段尾无效。

4. 弹性水平空白：`\hfill` 命令用于生成弹性水平空白，用于将当前行剩余的空间填满。如果当前行已经充满内容则无效。

`\hfill` 的这些衍生命令可以使用具体的形状而不是空白填充：

命令	效果
<code>\dotfill</code>	用点线填充
<code>\hrulefill</code>	用水平线段填充
<code>\downbracefill</code>	用开口向下的花括号填充
<code>\upbracefill</code>	用开口向上的花括号填充
<code>\leftarrowfill</code>	用向左的箭头填充
<code>\rightarrowfill</code>	用向右的箭头填充

效果示例：

```
chapter1.3 \dotfill filling \\  
left \leftarrowfill mid  
        \rightarrowfill right
```

```
chapter1.3 ..... filling  
left <-----mid ----->right
```

5. 自定义竖直空白：`\vspace{}` 用于在两段间产生宽度为零的竖直空白，参数为产生空白的竖直距离，负值表示拉近间距。

该命令置于页首或页尾无效。

带星号的 `\hspace*{}` 作用基本相同，但置于页首或页尾仍有效，一定能在其上下内容之间生成空白。

6. 定长竖直空白：`\smallskip` 生成一段高度为 $3\pm\frac{1}{4}$ 的可伸缩的垂直空白，输出时系统会自动选择范围内的一个值。类似地，`\medskip` 生成一段高度为 $6\pm\frac{2}{4}$ 的可伸缩的垂直空白；`\bigskip` 生成一段高度为 $12\pm\frac{4}{4}$ 的可伸缩的垂直空白。

7. 虚位竖直空白：`\vphantom{}` 在两段之间产生空白，但空白高度为参数中内容的竖直高度。它为参数的内容“预留竖直空白”。

该命令用在数学环境中，可主动调节 `\left` `\right` 的自适应大小。

8. 弹性竖直空白：`\vfill` 命令将当前页面剩余的垂直空间填满。该命令置于页首无效。

4.2 颜色

1. 简单的文本颜色: `\xcolor` 宏包提供了创建颜色的工具。`\color{}` 是一个声明, 它能使块中接下来所有文本(包括公式)变成参数的颜色, 例如声明 `\color{blue}` 会使接下来所有文本变为蓝色。
2. 页面颜色: 使用 `\pagecolor{}` 会更改当前及其之后页面的颜色, 在某一页使用 `\nopagecolor` 可以暂时去除该页面施加的颜色效果。

这是一个使用 `\pagecolor{black}` 和 `\color{white}` 的页面局部效果。

3. 颜色盒子: `\fcolorbox{ }{ }{ }` 可以创建一个颜色盒子, 为文本添加带有颜色的边框和背景, 三个参数分别是边框颜色、背景色和文本内容。

这是一段具有 blue 边框、yellow 背景的文本。

4. 局部颜色: 使用 `\textcolor{ }{ }` 可以创建局部的颜色, 两个参数分别为颜色名和影响的文本。它常用于公式环境中, 例如:

```
\[ \lim_{\textcolor{red}{\Delta}\rightarrow 0}
(1+\textcolor{red}{\Delta})^{\frac{1}{\Delta}} = e\]
```

$$\lim_{\Delta \rightarrow 0} (1 + \Delta)^{\frac{1}{\Delta}} = e$$

5. 颜色名: 可以直接使用颜色名指代颜色, `\xcolor` 宏包可以直接使用的颜色名约有 20 个。使用宏包的 `[dvipsnames]` 参数可以额外使用约 70 个颜色名, 例如 `BrickRed` 和 `Periwinkle` 这样的颜色名。

进一步使用宏包的 `[x11names]` 参数可以加载 300 多种颜色名, 例如 `Aquamarine3` 和 `Goldenrod2` 这样的颜色名。

6. 自定义颜色: 在导言区使用 `\definecolor{ }{ }{ }` 可以自定义颜色, 三个参数分别是自定义的颜色名、颜色格式、颜色值。颜色值的形式取决于颜色格式, 可用的颜色格式有:

- RGB: 红绿蓝混色, 每种色值取值范围为 0–255, 用逗号分隔
- rgb: 同上, 但每种色值取值范围为 0–1.0
- cmyk: 彩印标准颜色, 每种色值取值范围为 0–1.0
- gray: 灰度颜色, 取值范围为 0–1.0
- HTML: 六位十六进制 RGB 颜色值
- wave: 对应波长的颜色, 取值为 363–814

对于如下定义:

```
\definecolor{FrozenBlue}{RGB}{157,234,242}
\definecolor{CandleRed}{HTML}{F57267}
```

这是 `FrozenBlue` 和 `CandleRed` 的呈现效果。

如果 `\definecolor` 定义的颜色名和现有颜色名重复, 会覆盖之前定义的颜色名。`\providecolor` 命令用法类似, 但发现重复定义后会放弃新的定义。

7. 自定义混合颜色: `\colorlet{ }{ }` 提供了通过混合颜色来定义颜色的一种形式, 被混合颜色和混合比例均以感叹号 `!` 结尾, 例如:

```
\colorlet{PurplePink}{red!60!blue!40!}
```

会使用 60% 红色和 40% 蓝色混合出 `PurplePink` 颜色。

5 图表

5.1 图片

1. \LaTeX 中的图片：使用插图有两种途径，一是插入外部绘制的图片，二是使用 \LaTeX 代码直接在文档中画图。
2. 插入图片：需要 `\usepackage{graphicx}` 宏包支持，使用 `\includegraphics` 命令插图。可选参数调整图片属性，必选参数指明图形的路径：

```
Hello, \includegraphics
[height=0.5cm]{./LaTeX.png}
```

Hello, \LaTeX

支持的图形格式包括 PDF、PNG、JPG、EPS 等。

插入的图形就是一个有内容的矩形盒子，在正文中排版效果和一个很大的字符类似。

3. 图片环境：除了一些很小的标志图形，很少把插图直接夹在文字中。通常把图形放在一个可以变动相对位置的浮动环境中，使用 `figure` 环境：

```
\begin{figure}[ht]
  \centering
  \includegraphics{./LaTeX.png}
  \caption{\LaTeX symbol}
\end{figure}
```

\LaTeX

图 1: \LaTeX symbol

`figure` 环境有可选参数 `[ht]`，表示浮动体可以出现在环境周围的文本所在处 (here) 和一页的顶部 (top)。

`figure` 环境内部相当于默认没有缩进的段落。

`\caption` 命令给插图加上自动编号和标题。

5.2 表格

1. 基本表格结构：表格一般都直接用 \LaTeX 命令绘制。制作表格需要确定表格的行、列对齐模式和表格线，由 `tabular` 环境完成：

```
\begin{tabular}{|rrr|}
  \hline
    angle &  $\sin$  &  $\cos$  \\
  \hline
    0 & 0 & 1 \\
   $\pi/2$  & 1 & 0 \\
  \hline
\end{tabular}
```

angle	sin	cos
0	0	1
$\pi/2$	1	0

`tabular` 环境有一个参数，声明了表格中列的模式：`|rrr|` 表示表格有三列，都是右对齐 (right)，在第一列前面和第三列后面各有一条垂直的表格线。在 `tabular` 环境内部，行与行之间用换行命令 `\\` 隔开，每行内部的表项则用符号 `&` 隔开。表格中的横线则是用命令 `\hline` 产生的。

2. 表格的对齐：基本的对齐参数 `lcr` 表示左中右对齐，表格宽度会自动调整，但不会自动换行，可能超出页面宽度。可以用 `p{}` 形式指定某一列的宽度，参数内为宽度值，这时自动左对齐。
- 可以用可选参数表示文本垂直对齐方式，`tcb` 分别表示顶端对齐、垂直居中、底端对齐，默认顶端对齐。

3. 表格线：在必选参数内连续使用两个竖线 `||` 表示绘制一条竖直双线，使用 `@{}` 在每行对应绘制的竖直表格线替换为参数里的符号，参数为空表示不画竖直表格线。

命令 `\hline` 用于绘制一根水平表线，而命令 `\cline{c1-c2}` 仅绘制从 `c1` 列到 `c2` 列的水平表线。这两个命令后面不用加换行符，因为它算当前行的顶部。两个连续的 `\hline` 可以画水平双线，但是这种方式制造的双线与竖直表线的相交效果不好。

4. 竖向表线与拆分单元格：在单元格内使用 `\vline` 命令仅画出等于所在行行高的竖直线。该命令常常用于横向拆分单元格。

5. 表格的高级对齐：加载 `\array` 宏包可以使表格使用更多对齐效果，例如 `m{}` 在指定某列宽度时，也会使该行的其余部分垂直居中；`b{}` 在指定某列宽度时，也会使该行的其余部分底端对齐。例如：

	align	this column uses p{}	align	this column uses m{}	align	this column uses b{}
--	-------	-------------------------	-------	-------------------------	-------	-------------------------

6. 列格式：`\array` 宏包还提供了 `>{}` 和 `<{}`，分别用在 `lcrpmb` 参数前后，可以使该列每个单元格开头、结尾都套上参数内的命令或声明。例如以下表格参数：

<code>{ >\bfseries}c </code> <code>>\raggedleft\arraybackslash}p{6cm} }</code>	ohhh	It's a very very very long loong looong loooong loooooong line
---	------	---

使第一列单元格使用加粗字系，第二列单元格左边不平齐（就是右对齐）。其中 `\arraybackslash` 是为避免行末的 `\\` 出现异常。

7. 自定义列格式：可以在导言区通过自定义列格式命令 `\newcolumntype{f}{}` 将复杂的 `lcrpmb` 与 `>{}` 和 `<{}` 组合成一个格式，第一个参数是新的格式名（必须是单字母），第二个参数是组合格式。例如，在导言区定义：

```
\newcolumntype{f}{>{\(}c<{\)}}{}
```

则表格环境的参数中为 `f` 的列处于行内数学环境 `\(... \)` 并居中。

8. 小数对齐的列格式：宏包 `\siunitx` 提供了小数对齐的列格式，在导言区使用以下命令可以保留两位小数点：

```
\sisetup{
  round-mode=places,
  round-precision=2,
}
```

然后便可以使用 `S` 格式表示该列需要对齐小数点，效果为：

<pre>\begin{tabular}{ c S } \hline A & 1.234 \\ B & 12.34 \\ C & 123.4 \\ D & 12 \\ \hline \end{tabular}</pre>	<table> <tr><td>A</td><td>1.23</td></tr> <tr><td>B</td><td>12.34</td></tr> <tr><td>C</td><td>123.40</td></tr> <tr><td>D</td><td>12</td></tr> </table>	A	1.23	B	12.34	C	123.40	D	12
A	1.23								
B	12.34								
C	123.40								
D	12								

9. 表格与浮动：`tabular` 环境得到的也是一个比较大的盒子。一般也放在浮动环境 `table` 中，参数与大体的使用格式也与 `figure` 环境类似：

```
\begin{table}[H]
  % tabular
\end{table}
```

可选参数 `[H]` 表示不浮动 (Here)。该选项是由 `\float` 宏包提供的特殊功能。

10. 跨行列的单元格：需要 `\multirow` 宏包支持，分别使用 `\multirow{...}{...}` 和 `\multicolumn{...}{...}` 命令跨行和跨列。三个参数中：

- 第一个参数表示跨越（总共包含）的行数；
- 第二个参数同 `tabular` 环境的参数（对齐和画线），也可以使用 `*` 表示自动适应宽度；
- 最后一个参数是单元格内填入的文本。

跨行列后，被跨位置的单元格仍应正常表示且内容留空，它们会被覆盖以实现跨行列。

```
\begin{tabular}{cc}
  \hline
  \multirow{2}{*}{\multirow} & row1-2 \\
  & row2-2 \\
  \hline
\end{tabular}
```

	row1-2
multirow	row2-2

同时跨行跨列注意嵌套关系 `\multirow` 命令放在 `\multicolumn` 内部。

一个非常复杂的综合示例：

```
\begin{tabular}{|c|c|c|c|c|}
  \hline
  \multirow{2}{*}{\multi-row} & & & & \\
  \multicolumn{2}{c|}{\multi-col} & & & \\
  \multicolumn{2}{c|}{
    \multirow{2}{*}{\multi-row\&col}} \\
  \cline{2-3}
    & col-1 & col-2 & & \\
    \multicolumn{2}{c|}{} \\
  \hline
  item1&item2&item3&item4&item5 \\
  \hline
\end{tabular}
```

multi-row	multi-col		multi-row&col	
	col-1	col-2		
item1	item2	item3	item4	item5

`\cline{2-3}` 用于在 `multi-col` 下画一条第 2 栏位到第 3 栏位的边框线。空的 `\multicolumn` 用于修改同时跨行列单元格的边框线问题。

11. 拆分单元格：拆分单元格的需求可以通过表格嵌套实现。

12. 单元格换行：宏包 `\makecell` 提供了在单元格内换行方式，使用 `\makecell{...}` 命令可以在参数内使用 `\\` 方便地换行：

```
\begin{tabular}{|c|c|c|}
\hline col1 &
\makecell{line1 \\ line2} &
col3 \\ \hline
\end{tabular}
```

col1	row1	col3
	row2	

还可以配合可选参数使用 `[lcrbtb]` 之一指定单元格对齐方式。

13. 水平表线宽: `\makecell` 宏包还提供了 `\Xhline{}` 和 `\Xcline{ }{ }` 命令, 可以通过最后一个额外的参数指定水平表线宽。以下是一个模仿三线表的示例:

```
\begin{tabular}{ccc}
\Xhline{2pt}
\multirow{2}*{X} &
\multicolumn{2}{c}{Y} \\
\Xcline{2-3}{0.4pt} & Left & Right \\
\Xhline{1pt}
a & A & C \\
b & B & D \\
\Xhline{2pt}
\end{tabular}
```

X	Y	
	Left	Right
a	A	C
b	B	D

14. 分割表头: `\diagbox` 宏包提供了分割表头的命令 `\diagbox`。该命令支持两个或三个参数, 分别表示将表头分割成两部分或三部分, 例如:

```
\begin{tabular}{c|c}
\hline \diagbox{X}{data}{Y} & $y$ \\
\hline $x$ & 10 \\ \hline
\end{tabular}
```

data	Y	
X		y
x		10