

L^AT_EX is a system for typesetting scientific documents. The documents are written in plan text source and then *compiled* to produce a graphical output (as a PDF or an image). The document can contain formulae and figures, written in the L^AT_EX language, which are then rendered appropriately.

Regular text will be rendered as itself, but the following punctuation symbols have special meaning: The backslash symbol (\) is used for “commands” or “macros” which insert special symbols or notation into the text. Braces { and } are used to group symbols together into a block. Dollars (\$) and \$\$ are used to include formulae in the text.

There are two general modes of operation: “math mode” is used for formulae, and “text mode” is used for text. Formulae can be surrounded by single dollars to be included in the text “inline”, for example `$a + b = c$` produces: $a + b = c$. Double dollars render a large formula in “display style”, which inserts line breaks around the formula, and also has an effect on how some notation is rendered. For example `$$a + (b + c) = d$$` produces:

$$a + (b + c) = d$$

As may be evident, most symbols (namely !, ', (,), *, +, ,, -, ., /, :, ;, <, =, >, ?, [,], |) are rendered as themselves, however the commands in the following tables can be used to render other, more interesting kinds of symbols. To write literal braces and dollars, \{, \}, and \\$ can be used respectively.

To place a subscript or a superscript, _ and ^ can be used respectively. For example, `a~b` is a^b and `a_b` is a_b . To place more than one character in a sub- or superscript, the expression can be surrounded with { and }: `a~{b+c}` produces a^{b+c} .

The `\frac{}{}{}` command¹ renders a fraction (e.g. `\frac{a}{b}` is $\frac{a}{b}$), and `\sqrt{}{}` renders a square root (e.g. `\sqrt{a}` is \sqrt{a}). Order can be specified by writing e.g. `\sqrt[3]{a}` for $\sqrt[3]{a}$.

Sums are typeset with `\sum`, e.g. `\sum_{i=0}^n i^2` produces $\sum_{i=0}^n i^2$. In “display style”, subscripts and superscripts on `\sum` are rendered differently; the same formula produces:

$$\sum_{i=0}^n i^2$$

This behaviour is an example of a “big operator”. Others include `\prod`, `\lim`, `\bigcap`, etc.

Regular parentheses do not scale around a large expression, producing outputs like $(\frac{a}{b})$. Commands `\left` and `\right` can be used to produce a pair of parentheses (or other bracket-like symbols) that scales with the expression between them. The commands are followed by the type of bracket (like (or [), for example `\left(\frac{a}{b} \right)` produces $(\frac{a}{b})$. The `\left` and `\right` commands have to be balanced, but the exact bracket-like characters used don’t have to match, allowing for examples like $(-\infty, \frac{a}{b}]$.

In math mode, letters are *italicised* by default, as that is the convention for variable names. To typeset operation names in roman font, they should be put inside `\operatorname{...}`. Likewise, `\mathbb{...}` is used to render letters in the “blackboard” font, e.g. \mathbb{R} is produced by `\mathbb{R}`. Other available fonts include `\mathcal{...}` for calligraphic, `\mathscr{...}` for script, `\mathfrak{...}` for fraktur, and `\mathsf{...}` for sans-serif.

¹While it is definitely possible to write `\frac{1}{2}` for $\frac{1}{2}$, it is somewhat customary to always surround arguments to macros with braces.

Simple Algebra		Greek Letters	
<code>\div</code>	\div	<code>\alpha</code>	α
<code>\frac{a}{b}</code>	$\frac{a}{b}$	<code>\beta</code>	β
<code>\times</code>	\times	<code>\gamma</code>	γ
<code>a \cdot b</code>	$a \cdot b$	<code>\delta</code>	δ
<code>a^b</code>	a^b	<code>\Delta</code>	Δ
<code>a_b</code>	a_b	<code>\epsilon</code>	ϵ
<code>\pm</code>	\pm	<code>\varepsilon</code>	ε
<code>\mp</code>	\mp	<code>\zeta</code>	ζ
<code>\sqrt{a}</code>	\sqrt{a}	<code>\eta</code>	η
<code>\sqrt[b]{a}</code>	$\sqrt[b]{a}$	<code>\theta</code>	θ
<code>\neq, \not=</code>	\neq	<code>\vartheta</code>	ϑ
<code>\approx</code>	\approx	<code>\Theta</code>	Θ
<code>\sim</code>	\sim	<code>\iota</code>	ι
<code>\propto</code>	\propto	<code>\kappa</code>	κ
<code>\leq, \le</code>	\leq	<code>\lambda</code>	λ
<code>\geq, \ge</code>	\geq	<code>\Lambda</code>	Λ
<code>\ll</code>	\ll	<code>\mu</code>	μ
<code>\gg</code>	\gg	<code>\nu</code>	ν
<code>\leq</code>	\leq	<code>\pi</code>	π
<code>\lvert a \rvert</code>	$ a $	<code>\Pi</code>	Π
<code>\lfloor a \rfloor</code>	$\lfloor a \rfloor$	<code>\rho</code>	ρ
<code>\lceil a \rceil</code>	$\lceil a \rceil$	<code>\sigma</code>	σ
<code>\bar{a}</code>	\bar{a}	<code>\Sigma</code>	Σ
<code>\Re</code>	\Re	<code>\tau</code>	τ
<code>\Im</code>	\Im	<code>\upsilon</code>	υ
<code>a \circ b</code>	$a \circ b$	<code>\Upsilon</code>	Υ
<code>\mathbb{N}</code>	\mathbb{N}	<code>\phi</code>	ϕ
		<code>\varphi</code>	φ
		<code>\Phi</code>	Φ
Set Theory & Logic			
<code>\in</code>	\in	<code>\chi</code>	χ
<code>\notin</code>	\notin	<code>\psi</code>	ψ
<code>\varnothing</code>	\varnothing	<code>\Psi</code>	Ψ
<code>\subset</code>	\subset	<code>\omega</code>	ω
<code>\subseteq</code>	\subseteq	<code>\Omega</code>	Ω
<code>\supset</code>	\supset		
<code>\supseteq</code>	\supseteq		
<code>\cup</code>	\cup		
<code>\cap</code>	\cap		
<code>\setminus</code>	\setminus		
<code>\forall</code>	\forall		
<code>\exists</code>	\exists		
<code>\implies</code>	\implies		
<code>\iff</code>	\iff		
Simple Geometry		Calculus	
<code>\parallel</code>	\parallel	<code>\sum</code>	\sum
<code>\nparallel</code>	\nparallel	<code>\prod</code>	\prod
<code>\perp</code>	\perp	<code>\coprod</code>	\coprod
<code>\angle</code>	\angle	<code>\infty</code>	∞
<code>\triangle</code>	\triangle	<code>\rightarrow</code>	\rightarrow
<code>\square</code>	\square	<code>\mapsto</code>	\mapsto
<code>\overrightarrow{AB}</code>	\overrightarrow{AB}	<code>\uparrow</code>	\uparrow
<code>\overline{AB}</code>	\overline{AB}	<code>\downarrow</code>	\downarrow
		<code>\prime</code>	\prime
		<code>\partial</code>	∂
		<code>\dot{a}</code>	\dot{a}
		<code>\ddot{a}</code>	\ddot{a}
		<code>\int_a^b</code>	\int_a^b
		<code>\iint</code>	\iint
		<code>\iiint</code>	\iiint
		<code>\oint</code>	\oint
		<code>\nabla</code>	∇