Introduction to LaTeX

Week 05: Theorems, Algorithms, Code, Units

2026-11-12

Table of Contents

1. Theorems and Proofs
2. Algorithms
3. Source Code
4. Numbers and Units
5. Commands of Your Own
6. Errors and Obsolete Practices

Theorem-Like Environments: \newtheorem

Input

```
\usepackage{amsthm}
\newtheorem{theorem}{Theorem}[section]
\begin{document}
\section{Numbers}
\begin{theorem}[Fermat]
  \label{theorem:fermat}
  There are no positive integers $x, y, z$
  with $x^n + y^n = z^n$ for any $n \ge 3$.
\end{theorem}
```

Output

1 Numbers

Theorem 1.1 (Fermat)**.** *There are no positive integers* x, y, z *with* $x^n + y^n = z^n$ *for any* $n \geq 3$.

- `\newtheorem` defines the environment `theorem` with the heading "Theorem".
- `[section]` numbers theorems within sections: 1.1, 1.2, 2.1, ...
- The optional argument of `\begin{theorem}` adds a note, such as a name.

Using the amsthm Package

One Counter for All

Definitions of theorem-like environments

```
\newtheorem{theorem}{Theorem}[section]
\newtheorem{lemma}[theorem]{Lemma}
\newtheorem{corollary}[theorem]{Corollary}
```

Possible output

Lemma 1.1. ...
Theorem 1.2. ...
Corollary 1.3. ...

- `[theorem]` between the braces: `lemma` uses the counter of `theorem`.
- Subordinate environments then follow the same numbering scheme as the main (`theorem`) environment. That is, lemmas and corollaries are numbered within sections, too.

One counter or several?

With one counter, the numbers show the order: Lemma 2.4 comes after Theorem 2.3. With separate counters, Theorem 1 and Lemma 1 may be pages apart, and readers have to search.

Theorem Styles

Definitions of theorem-like environments

```
\theoremstyle{plain} % the default
\newtheorem{theorem}{Theorem}[section]
\newtheorem*{theorem*}{Theorem}

\theoremstyle{definition}
\newtheorem{definition}[theorem]{Definition}
\newtheorem*{definition*}{Definition}

\theoremstyle{remark}
\newtheorem{remark}[theorem]{Remark}
\newtheorem*{remark*}{Remark}
```

Look of the styles

`plain`:
bold heading, italic text

`definition`:
bold heading, upright text

`remark`:
italic heading, upright text

- `\theoremstyle` applies to all `\newtheorem` that follow it.
- `plain` suits theorems, lemmas and corollaries; `definition` suits definitions, examples and assumptions; `remark` suits remarks and notes.
- `\newtheorem*` defines an environment without a number.

Theorem Styles in the Document

In the document

```
\section{Sequences}
\begin{definition}
  A sequence is bounded if \dots
\end{definition}
\begin{theorem}
  Every bounded sequence
  has a convergent subsequence.
\end{theorem}
\begin{remark*}
  The converse does not hold.
\end{remark*}
```

Output

1 Sequences

Definition 1.1. A sequence is bounded if ...

Theorem 1.2. *Every bounded sequence has a convergent subsequence.*

Remark. The converse does not hold.

- Definition and theorem share one counter: the numbers follow the order of use.
- `remark*` gives a remark without a number, as `equation*` gives a display without one.

Proofs: proof

Input

```
\begin{proof}
  The claim follows by induction.
\end{proof}
\begin{proof}[Proof of the main theorem]
  For all $y$, we have
  \begin{equation*}
    f(y) \ge f(x^*).
    \qedhere
  \end{equation*}
\end{proof}
```

Output

Proof. The claim follows by induction. □

Proof of the main theorem. For all y, we have

$$f(y) \geq f(x^*).$$ □

- `proof` sets the heading "Proof." and the end mark □.
- When a proof ends with a display, `\qedhere` puts the end mark into its last line.
- The optional argument of `\begin{proof}` replaces the heading.

Referring to Theorems: \zcref

Input	Output
`by \zcref{theorem:fermat}`	by theorem 1.1
`by \zcref{lemma:gradient,theorem:minimizer}`	by lemma 2.1 and theorem 2.2
`\zcref[S]{theorem:minimizer} shows`	Theorem 2.2 shows

- `zref-clever.sty` names a reference after the counter. A lemma that uses the counter of `theorem` would therefore be called "theorem".
- The course package `latex-course.sty` corrects this for `lemma`, `proposition`, `corollary`, `definition`, `example` and `remark`, with one line each:

```
\AddToHook{env/lemma/begin}
  {\zcsetup{countertype = {theorem = lemma}}}
```

Your Own Theorem-Like Environment: \zcRefTypeSetup

Input

```
\theoremstyle{definition}
\newtheorem{assumption}[theorem]{Assumption}

\AddToHook{env/assumption/begin}
  {\zcsetup{countertype = {theorem = assumption}}}

\zcRefTypeSetup{assumption}{
  Name-sg = Assumption, name-sg = assumption,
  Name-pl = Assumptions, name-pl = assumptions,
}
```

\zcref to an assumption gives

without both lines:
theorem 1.2

without \zcRefTypeSetup:
?? 1.2

with both lines:
assumption 1.2

- The hook tells `zref-clever.sty` the type of the reference: `assumption`.
- \zcRefTypeSetup tells it how to name that type, in the singular and the plural, capitalized or not.
- `zref-clever.sty` knows names for theorem, lemma, corollary, proposition, definition, proof, remark, example and a few more.

Table of Contents

1. Theorems and Proofs
2. Algorithms
3. Source Code
4. Numbers and Units
5. Commands of Your Own
6. Errors and Obsolete Practices

Pseudocode: The algorithmic Environment

Input

```
\begin{algorithmic}[1]
  \Procedure{Bisection}{$f, a, b, \varepsilon$}
    \While{$b - a > \varepsilon$}
      \State $m \gets (a + b) / 2$
      \If{$f(a) \, f(m) \le 0$}
        \LComment{A root lies in $[a, m]$.}
        \State $b \gets m$
      \Else
        \State $a \gets m$
      \EndIf
    \EndWhile
    \State \Return $(a + b) / 2$
  \EndProcedure
\end{algorithmic}
```

Output

1: **procedure** BISECTION(f, a, b, ε)
2: **while** $b - a > \varepsilon$ **do**
3: $m \leftarrow (a + b)/2$
4: **if** $f(a)\, f(m) \le 0$ **then**
5: ▷ *A root lies in* $[a, m]$. ◁
6: $b \leftarrow m$
7: **else**
8: $a \leftarrow m$
9: **return** $(a + b)/2$

- `\usepackage{algpseudocodex}` provides the `algorithmic` environment.
- `[1]` numbers every line.

algpseudocodex Package Documentation

Pseudocode: Block Lines and noEnd

Default

```
1: procedure Bisection(f, a, b, ε)
2:     while b − a > ε do
3:         m ← (a + b)/2
4:         if f(a) f(m) ≤ 0 then
5:             ▷ A root lies in [a, m]. ◁
6:             b ← m
7:         else
8:             a ← m
9:     return (a + b)/2
```

noEnd = false, indLines = false

```
1: procedure Bisection(f, a, b, ε)
2:     while b − a > ε do
3:         m ← (a + b)/2
4:         if f(a) f(m) ≤ 0 then
5:             ▷ A root lies in [a, m]. ◁
6:             b ← m
7:         else
8:             a ← m
9:         end if
10:    end while
11:    return (a + b)/2
12: end procedure
```

- By default, lines show where a block ends. `\usepackage[noEnd = false, indLines = false]{algpseudocodex}` shows lines such as "end while" instead.
- `\LComment{...}` sets a long comment on a line of its own; `\Comment{...}` sets a short one at the right margin.

Table of Contents

1. Theorems and Proofs
2. Algorithms
3. Source Code
4. Numbers and Units
5. Commands of Your Own
6. Errors and Obsolete Practices

Source Code in the Document: `minted.sty`

Input

```
\usepackage{minted}
\begin{document}
\mintinline{python}{square(3)} returns 9.
\begin{minted}[linenos]{python}
def square(x):
  """Return the square of x."""
  return x * x
\end{minted}
```

Output

`square(3)` returns 9.

```
1  def square(x):
2    """Return the square of x."""
3    return x * x
```

- `\mintinline` sets code within a sentence, the environment `minted` sets a block of code.
- The first argument names the language; `minted.sty` knows several hundred. Options such as `[linenos]` for line numbers go before it.

Source Code from a File: \inputminted

Input

```
\inputminted{python}{squares.py}
```

Output

```python
# Print a few squares.
def square(x):
  """Return the square of x."""
  return x * x

for i in range(1, 4):
  print(i, square(i))
```

- The code stays in a file of its own.
- Changes to the file show up at the next compilation.
- `\inputminted[firstline = 2, lastline = 4]{python}{squares.py}` shows part of a file.

How `minted.sty` Works

- `minted.sty` passes the code to Pygments, a Python program, and reads the highlighted result back. The helper program `latexminted` does this.
- Running a program from LaTeX is called shell escape. By default, it is restricted to a list of trusted programs. In TeX Live 2024 and later, and on Overleaf, this list includes `latexminted`, so `minted.sty` works without further setup.
- The results are kept in a folder `_minted`, which can be deleted at any time.

Code without colors?

If the log says `minted v3+ executable is not installed`, the helper program is missing or too old for your Python. Install it with `pip install latexminted`, which brings Pygments along, update your TeX installation, or use Overleaf.

Full Shell Escape

- Older TeX Live versions do not trust `latexminted` and need full shell escape:

  ```
  pdflatex -shell-escape paper.tex
  latexmk -pdf -shell-escape paper.tex
  ```

- `latexmk` passes the option on to `pdflatex` at every run.
- MiKTeX calls the option `-enable-write18`.
- The log shows which kind was active: `restricted \write18 enabled.` or `\write18 enabled.`

When to allow full shell escape

Full shell escape lets a document run any program on your computer. Allow it only for documents you trust, and keep the restricted default otherwise.

An Older, Less Powerful Alternative to `minted.sty`: `listings.sty`

Input

```
\usepackage{listings}
\lstset{language = Python}
\begin{document}
\begin{lstlisting}
def square(x):
  """Return the square of x."""
  return x * x
\end{lstlisting}
```

Output

```
def square(x):
  """Return the square of x."""
  return x * x
```

- `listings.sty` needs no external program: TEX itself picks out keywords, comments and strings.
- It knows fewer languages, and it shows no colors until you choose them.

Table of Contents

1. Theorems and Proofs
2. Algorithms
3. Source Code
4. Numbers and Units
5. Commands of Your Own
6. Errors and Obsolete Practices

What siunitx.sty Offers

Command	Sets
\num	numbers: digit groups, minus signs, exponents, uncertainties
\unit	units: upright, with prefixes, powers and fractions
\qty	quantities: a number and its unit, with the right space between
\numlist, \numrange	lists and ranges, also of quantities
\ang	angles in degrees, minutes and seconds
\complexnum	complex numbers
column type S	numbers in tables, aligned at the decimal marker

- siunitx.sty follows the rules of the International System of Units (SI): units upright, a space between number and unit.
- \sisetup changes the output for the whole document, such as a decimal comma, while the input stays the same.

Numbers: \num

Input	Output
`\num{12345.678}`	$12\,345.678$
`\num{-1.23e-4}`	-1.23×10^{-4}
`\num[exponent-product = \cdot]{-1.23e-4}`	$-1.23 \cdot 10^{-4}$
`\num{1.23(4)}`, `\num{1.23 +- 0.04}`	$1.23(4)$

- `\num` works in text and in math.
- It groups the digits of long numbers, sets a proper minus sign and writes exponents as powers of ten.
- An uncertainty may be given in parentheses or with +-.
- `\sisetup{exponent-product = \cdot}` does the same for the whole document.

Units and Quantities: \unit and \qty

Input	Output
`\unit{\kilo\gram}`	kg
`\qty{9.81}{\meter\per\second\squared}`	$9.81\,\mathrm{m\,s^{-2}}$
`\qty{9.81}{m/s^2}`	$9.81\,\mathrm{m/s^2}$
`\qty{20}{\degreeCelsius}`	$20\,^{\circ}\mathrm{C}$

- `\qty` joins number and unit with the right space and without a line break between them.
- Units are set upright, in text and in math alike.
- Units may also be typed literally, as in the last line.

Percentages, Ranges and Lists

Input	Output
`\qty{25}{\percent}`	$25\,\%$
`\numrange{1}{10}`	1 to 10
`\qtyrange{10}{20}{\degreeCelsius}`	$10\,^{\circ}\mathrm{C}$ to $20\,^{\circ}\mathrm{C}$
`\qtylist{1;2;3}{\meter}`	$1\,\mathrm{m}$, $2\,\mathrm{m}$ and $3\,\mathrm{m}$
`\qtyproduct{2 x 3}{\meter}`	$2\,\mathrm{m} \times 3\,\mathrm{m}$

- `\percent` treats a percentage as a quantity, with a thin space before the sign.
- Ranges, lists and products repeat the unit after every number.

One Setting for the Whole Document: \sisetup

Setting	Without	With
`per-mode = symbol`	$9.81\,\mathrm{m\,s^{-2}}$	$9.81\,\mathrm{m/s^2}$
`per-mode = fraction`	$9.81\,\mathrm{m\,s^{-2}}$	$9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$
`round-mode = places, round-precision = 2`	3.141 59	3.14
`output-decimal-marker = {,}`	3.14	3,14

- `\sisetup{...}` in the preamble applies to every number and unit in the document.
- The same keys work for a single command:
  `\qty[per-mode = symbol]{9.81}{\meter\per\second\squared}`.

Table of Contents

1. Theorems and Proofs
2. Algorithms
3. Source Code
4. Numbers and Units
5. Commands of Your Own
6. Errors and Obsolete Practices

The Same Thing, Typed Forty Times

Input

```
Let $\Omega \subseteq \mathbb{R}^d$
and $f \colon \Omega \to \mathbb{R}$.
We seek $u \colon \Omega \to \mathbb{R}$
with $\norm{u}_{H^1(\Omega)}
  \le \norm{f}_{L^2(\Omega)}$.
...
For $\alpha \in \mathbb{R}$
and $\Omega \subseteq \mathbf{R}^d$, ...
```

Output

Let $\Omega \subseteq \mathbb{R}^d$ and $f \colon \Omega \to \mathbb{R}$.
We seek $u \colon \Omega \to \mathbb{R}$ with
$\|u\|_{H^1(\Omega)} \le \|f\|_{L^2(\Omega)}$. ...
For $\alpha \in \mathbb{R}$ and $\Omega \subseteq \mathbf{R}^d$, ...

- Operators and paired delimiters have names already, through `\DeclareMathOperator` and `\DeclarePairedDelimiter`.
- Everything else, such as $\mathbb{R}$ or a norm with a fixed subscript, is typed again each time.
- Sooner or later, one occurrence differs.

Commands of Your Own: \newcommand

Input

```
\newcommand{\R}{\mathbb{R}}
\newcommand{\normH}[1]{\norm{#1}_{H^1(\Omega)}}
\begin{document}
Let $f \colon \Omega \to \R$.
Then $\normH{u}
  \le \norm{f}_{L^2(\Omega)}$.
```

Output

Let $f \colon \Omega \to \mathbb{R}$. Then
$\|u\|_{H^1(\Omega)} \le \|f\|_{L^2(\Omega)}$.

- `\newcommand` goes into the preamble.
- `[1]` says that the command takes one argument; `#1` stands for it in the definition.
- To change the notation, change the definition: every occurrence follows.

Choosing Names

- Name what a thing means, not how it looks: `\asVector{x}` rather than `\bfx`. Switching from bold vectors to arrows then takes one line.
- Names consist of letters only: `\R2` is `\R` followed by 2.
- `\newcommand` refuses names that exist already, such as `\span`. Choose another name, such as `\linspan`, rather than overwriting the existing command.

When to define a command

Define a command when you type the same notation for the third time, or when you may want to change the notation later.

Table of Contents

1. Theorems and Proofs
2. Algorithms
3. Source Code
4. Numbers and Units
5. Commands of Your Own
6. Errors and Obsolete Practices

Errors You Will Meet

Message	Usual cause
`! LaTeX Error: Environment theorem undefined.`	`\newtheorem` missing
`! LaTeX Error: Command \span already defined.`	the name is taken; choose another
`! Illegal parameter number in definition of \normH.`	`#1` without `[1]`
`! Undefined control sequence.` “`\State`”	`algpseudocodex.sty` not loaded
`! Package siunitx Error: Invalid number '3.1x'.`	letters in `\num`
`! Package minted Error: minted v3+ executable is not installed`	helper program missing or too old

Mistakes Without an Error Message

Mistake	Effect
a lemma with the counter of `theorem`, without the hook	`\zcref` calls it "theorem"
a proof ending with a display, without `\qedhere`	the end mark stands on a line of its own
`\num{1,000}`	1.000: the comma is read as the decimal marker
`$3 m/s$`	$3m/s$: the unit set in italics, as variables
-5 in text	-5: a hyphen instead of a minus sign

Some Obsolete Practices

Obsolete	Use instead	Why
`\textit{Proof.} ... $\square$`	`proof`	the end mark is placed for you
verbatim for code	minted	highlighting, line numbers, code from files
`9.81\,$\mathrm{m}/\mathrm{s}^2$`	`\qty`	consistent spacing, no line break inside
`\SI{9.81}{\meter}`, `\si{\meter}`	`\qty`, `\unit`	the names of `siunitx.sty` version 2
`\def\R{\mathbb{R}}`	`\newcommand`	`\def` overwrites existing commands without a word

References

- American Mathematical Society (2017). *Using the amsthm Package*. Version 2.20.3. URL: https://ctan.org/pkg/amsthm.
- Barros, G. (2024). *The zref-clever Package. User Manual*. Version 0.5.1. URL: https://ctan.org/pkg/zref-clever.
- Matt, C. (2025). *algpseudocodex Package Documentation*. Version 1.2.0. URL: https://ctan.org/pkg/algpseudocodex.
- Poore, G. M. (2025). *The minted Package. Highlighted Source Code in LaTeX*. Version 3.7.0. URL: https://ctan.org/pkg/minted.
- Wright, J. (2025). *siunitx. A Comprehensive (SI) Units Package*. Version 3.4.14. URL: https://ctan.org/pkg/siunitx.