

# xlistings v1.0.0

## opinionated listings extensions

Florian Sihler\*

August 17, 2026

## 1 Introduction

This package extends on the `listings` package, providing an easier front-end to create code blocks of selected languages, support for number highlighting, segment highlighting, non-selectable line numbers,<sup>1</sup> and more. This package is not compatible with the `minted` package. In summary this package provides the following improvements over the `listings` package:

- Highlighting of numbers in code blocks: `10_234 + x1 * 0x34 - x2` (`hlnumbers` and `extendednums` options)
- Support for the `\begin{minted}<lang> . . . \end{minted}` environment (`fakeminted` option)
- Wrapper macros like `\bjava{int i}` and `\cjava{int i}` and environments `\begin{plainjava}`
- Language sensitive override: `\xlstlangoverride{latex}{morekeywords=[5]{\xlstsetstyle}}`
- Support for (`accsupp` based) non-selectable line numbers and characters:

```
1 System.out.println("Hello, World!");
```

- Support for blacklisting line numbers with `\xlstblacklistlinenumbers`
- Support for umlauts and UTF-8 encoding (with the `listingsutf8` package)
- Provides autogobble to remove leading spaces (with the `lstaugobble` package)
- Comfort key add to `literate` to add elements to the `literate` list
- `\LoadLanguages<lang>` to load a language or multiple languages on demand
- Opinionated language overwrites (see the `langs/` folder)
- Opinionated default literates such as `:ldots: ( . . . )`, `:lan: ( ( )`, `:to: ( → )`, and `:c: ( , breaks highlighting)`
- `\BadgeNextListing<lang>` to show a language tag at the start of the next listing (paragraph). The macros `\xlstmintedwithlangbadge` and `\xlstmintedwithoutlangbadge` toggle this behavior for all `fakeminted` environments.<sup>2</sup>

### 1.1 Loading the Package

To load the package, simply use:

```
1 \usepackage{xlistings} latex
```

\*Released under the [LaTeX Project Public License](#), version 1.3c or later.

<sup>1</sup>If a number is truly non-selectable depends on the viewer used. To ensure that they can not be selected would require images, a process we currently do not support.

<sup>2</sup>If you want to change the style of these badges, redefine the `\xlstlangbadgestyle` macro which receives the name of the language as its first argument. This badge will be automatically unselectable if `accsupp` is available.

Throughout this document, we will use the `xlistings` package to highlight code. If you are no fan of the “bordered” look of the code blocks, you can use `\xlstsetstyle{plain number}` to switch the look and feel, or use the `style` option of the package:

```
1 \usepackage[style={plain number}]{xlistings} latex
```

Or, using `\xlstsetstyle{plain}`:

```
\usepackage[style=plain]{xlistings} latex
```

## 1.2 Origin

Originally, this package was part of [LILLY](#), which I again ported to the [sopra-collection](#) during my studies. This package is the standalone and improved version of these packages. The process is work in progress and I welcome any kind of feedback.

## 1.3 Dependencies

This package imports the following packages:

- |                                                                  |                                                              |
|------------------------------------------------------------------|--------------------------------------------------------------|
| • <code>kvoptions</code>                                         | • <code>lstautogobble</code>                                 |
| • <code>needspace</code> <sup>(<a href="#">guardspace</a>)</sup> | • <code>listingsutf8</code> <sup>(<a href="#">try</a>)</sup> |
| • <code>etoolbox</code>                                          | • <code>listings</code>                                      |
| • <code>xcolor</code>                                            | • <code>accsupp</code> <sup>(<a href="#">try</a>)</sup>      |
| • <code>pgfkeys</code>                                           | • <code>tikz</code> <sup>(<a href="#">highlights</a>)</sup>  |

*“try” notes that the package is only loaded if present and may rely on a fallback if not. For example, `accsupp` allows to make line numbers “uncopyable” but is not critical so if it is not found we provide fallbacks.*

All of those package should be part of usual  $\text{\LaTeX}$  distribution.

## 2 References

### 2.1 Accepted Parameters

`print` *package option*

This option tries to save ink by not highlighting the background of code blocks and by making certain keywords bold or italic to ensure readability even in monochrome prints. See also [digital](#) for the complementary option.

`digital` *default, package option*

This is the complementary option to [print](#) and is the default.

`guardspace` *default, package option*

This option allows you to configure a minimum number of lines that should be kept together, which you can define with the help of `\xlstguard`. If you have a border this can help avoid initial stripes.

If you explicitly want to disable this option, use `guardspace=false` when loading the package.

`numinpar` *package option*

This option is a shorthand for `\xlstSetLeftMargin` with the value “2em”. It can be used to automatically indent listings and prevent numbers from protruding beyond the document margin.

`hlnumbers` *default, package option*

This option causes the indicative number highlighting in code blocks and is on by default, use `hlnumbers=false` to disable it.

`upshape` *package option*

By default, our inline commands adapt to the surrounding text and may be e.g., italic. This option disables this behavior enforcing an upshape style. For a similar behavior but with font size, see [inlinesize](#)

`inlinesize` *package option*

By default, our inline commands adapt to the surrounding fontsize, this option disables this behavior enforcing a fixed size. For a similar behavior but with font shape, see [upshape](#). to configure the font size, see `\lstfs`.

`highlights` *package option*

This option uses the [tikz](#) package to highlight code segments. This can be used to highlight code segments in a different color or with a background. This option is currently work in progress and not easily usable.

`fakeminted` *default, package option*

This option causes `xlistings` to provide you with a replacement minted environment. A language that is not loaded yet is loaded on demand, so `\begin{minted}<lang>` works without a preceding `\LoadLanguages`; if no definition can be found, you get an error naming the language. To disable this behavior, use `fakeminted=false`.

`extendednums` *package option*

This option extends the number highlighting beyond plain decimals:

- digit separators: `10_234`
- the classic `0x-`, `0b-` and `0o-` prefixes (upper case variants included), where the prefix itself uses the numprefix style: `0x1A`, `0b1010`, `0o17`
- the letters of a hexadecimal literal: `0xdeadBEEF`
- exponents and type suffixes: `1e10`, `1.5f`, `10L`

A letter is only ever taken as part of a number if it directly follows a highlighted number, so `data1` and `x0xFF` stay untouched.

Which letters may end a number depends on the language and is configured with the `listings` key `number suffixes`, which defaults to `eE`: `10L` is a long in Java, `3j` a complex number in Python, `10n` a `BigInt` in TypeScript, and in prose `3d` is just a “d” behind a three. A language switches the letters off completely with `number suffixes={}`.

`debug` *package option*

This option enables debug output and causes the package to print debug information to the console.

`style` *string, package option*

This option selects the default style that can be set with `\xlstsetstyle`. The default is (surprise) `default`.

## 2.2 Most Important Commands

After loading the package as described in section [1.1](#), you can load a language with `\LoadLanguages`:

```
\LoadLanguages{latex, java, bash, json}
```

latex

This uses the opinionated language definitions in the `langs/` folder and provides them to you in a set of environments and commands:

- As `\begin{<lang>}`, `\begin{<lang>*`, and `\begin{plain<lang>}` environments. For example, with the before loaded languages you have:

```
\begin{java}                                     latex
...
\end{java}
```

- As `\c<lang>{<code>}` and `\b<lang>{<code>}` commands, e.g., `\cjava{int i}` and `\bjava{int i}`.
- As `\i<lang>{<file>}` command to include a file.
- As `\lstc{<lang>}{<code>}`, `\lstb{<lang>}{<code>}`, and `\lsti{<lang>}{<file>}`, which take the language as an argument. This is the only way to reach a language whose name cannot be part of a command name, as `\bx86{<code>}` would be read as `\bx` followed by “86”.
- As `\begin{minted}{<lang>}` environment.

All inline commands (`\c<lang>`, `\b<lang>`, `\i<lang>`) have an advantage in that they can be used as arguments to other macros, *but* you have to escape things like backslashes or curly braces with an additional backslash.

If you want to register special keywords or any other configuration for an existing, and loaded, language, you can use `\xlstlangoverride`:

```
1 \xlstlangoverride{latex}                        latex
2     morekeywords={5}{\commandA, \commandB},
3     lineskip=42pt,
4     % ...
5 }
```

As you can see, the sample above skips a given line number, which we achieve by using the following command: `\lstblacklistlinenumbers{4}`. The effects of all of these commands are local, so you can use them in a group (e.g., with `\begingroup` and `\endgroup` or an opening and closing brace) to limit their effect.

Additionally, if you look at the source of this documentation, the source of the example above is actually indented. We remove the leading spaces with the new `autogobble` option (provided by the `lstautogobble` package). If you want to add new literates for whatever reason, you can use the new `listings` key `add to literate`:

```
1 \lstset{                                         latex
2     add to literate={sample}{\(\mathcolor{red}{\pi}\)}{1}
3 }
```

With this active, writing sample will result in  $\pi$  (you can combine this with `\xlstlangoverride` to add literates to a specific language).

If you want to change the colors of the keywords used, you can use `\lstcolorlet` with the keys `keywordA`, `keywordB`, `keywordC`, `numbers`, `numprefix`, `literals`, `comments`, `highlight`, `command`, and `linenumbers`:

```
1 \lstcolorlet{comments}{orange}                 latex
```

With this, the color of comments will be changed to orange:

```
1 % sample comment                               latex
```

If you want to have complete control over the styling, you may use `\xlstDefineStyles` which works like the following:

```
1 \xlstDefineStyles{%                             latex
2     {comments: \color{green}\scshape},%
3 }
```

Now, comments will be displayed in green and bold:

```
1 % sample comment
```

latex

*This command is currently rather rigid and subject to change!*

If you are interested in a flexible highlighting and animation of these code snippets, have a look at the [code-animation](#) package, which we are currently working for to be compatible with xlistings.

## 2.3 Writing a Language

A language lives in a file `xlistings-<name>.cfg` which is picked up by `\LoadLanguages` and consists of an ordinary `\lstdefinlanguage` followed by `\RegisterLanguage`:

```
1 \lstdefinlanguage{lRubberduck}{
2     comment=[1]{\#},
3     morekeywords = {Quack, new},
4     morekeywords = [2]{Duck}
5 }
6 \RegisterLanguage{rubberduck}{lRubberduck}
```

latex

There are three things worth knowing when you write such a file:

- The name of the listings language starts with an “l” followed by an upper case letter (lRubberduck) so that it never collides with a language that listings ships itself. The name you register (rubberduck) is the one used for the environments, the inline commands, the minted environment, and `\xlstlangoverride`.
- Use only the public macros of this package. Language files are read when @ is not a letter, so `\xlst@`. . . macros are not available; `\xlstGet` and `\xlstGetStyle` are.
- All languages share one column model (`columns=flexible`), set by the listing style. Only override it if the language really needs something else.
- The keyword classes [1] to [6] are styled as keywordA, keywordB, keywordC, keywordD, literals, and command. A language may of course set its own `keywordstyle`.

The behavior of a language is pinned down by the test suite in `tests/`: `l3build check states`, per sample, which styles are expected in which order, and `build/test/<name>.pdf` shows the verdict next to the typeset sample so the result can be inspected by eye. A color that leaks past its segment does not show up in such a trace, so the checks additionally compare the pdf, reduced to the color every piece of text is drawn in.

For segment highlighting, the macros `\xlsthllwarning`, `\xlsthllerror`, and `\xlsthllinfo` are available as delimiter styles (they fall back to a plain color if the [highlights](#) option is not active), as are `\xlsthllbox{<color>}` and `\xlsthllboxd{<tikz keys>}`.

## 2.4 Language Samples

Every language shipped with this package is listed below, tagged with the name you pass to `\LoadLanguages`. They are opinionated: they favor the constructs the author needs and can be adapted with `\xlstlangoverride`.

```
1 public static void main(String[] args){
2     // This is a comment
3     int number = 10_234 + 0x1A - 0b1010;
4     System.out.println("Hello, World!"); // Print greeting
5 }

1 aspect Logging {
2     pointcut trace(): execution(* *.*(..));
3 }
```

java

aspectj

|                                                                                                                                                                                                        |            |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| 1 <code>int main(void) { /* a comment */</code><br>2 <code>printf("%d\n", 0xdeadBEEF);</code><br>3 <code>}</code>                                                                                      | cpp        |
| 1 <code>public static int Add(int a, int b) =&gt; a + b; // null, true, false</code>                                                                                                                   | csharp     |
| 1 <code>def area(r): # a comment</code><br>2 <code>return 3.14159 * r ** 2 if r &gt; 0 else None</code>                                                                                                | python     |
| 1 <code>squares &lt;- sapply(c(1, 2, 3.45), function(x) x^2) # a comment</code>                                                                                                                        | R          |
| 1 <code>const answer = async () =&gt; { return 0x2A; }; // a comment</code>                                                                                                                            | javascript |
| 1 <code>function id&lt;T&gt;(value: T): T { return value; } // a comment</code>                                                                                                                        | ts         |
| 1 <code>double :: Int -&gt; Int -- a comment</code><br>2 <code>double x = 2 * x</code>                                                                                                                 | haskell    |
| 1 <code>local t = {1, 2, 3} -- a comment</code><br>2 <code>for i, v in ipairs(t) do print(i, v) end</code>                                                                                             | lua        |
| 1 <code>ls -la /tmp   grep tex # a comment</code><br>2 <code>sudo apt-get install texlive</code>                                                                                                       | bash       |
| 1 <code>git commit -m "message"</code><br>2 <code>git push origin master</code>                                                                                                                        | git        |
| 1 <code>SELECT name FROM users WHERE id &gt; 10; -- a comment</code>                                                                                                                                   | sql        |
| 1 <code>{ "name": "xlistings", "version": 1, "tags": [true, false, null] }</code>                                                                                                                      | json       |
| 1 <code># This is a comment</code><br>2 <code>key: value</code><br>3 <code>list:</code><br>4 <code>- item1</code><br>5 <code>- item2: subvalue</code><br>6 <code>number: 0x1A + 10_000 - 0b1010</code> | yaml       |
| 1 <code>&lt;!-- a comment --&gt;</code><br>2 <code>&lt;book id="1"&gt;&lt;title&gt;Listings&lt;/title&gt;&lt;/book&gt;</code>                                                                          | xml        |
| 1 <code># Heading</code><br>2 <code>A <b>bold</b> word and some <code>`code`</code>.</code>                                                                                                            | markdown   |
| 1 <code>\documentclass{article}</code><br>2 <code>\begin{document}</code><br>3 <code>\ldots</code><br>4 <code>\end{document}</code>                                                                    | latex      |
| 1 <code>^(0x)?[0-9a-fA-F]+\$</code>                                                                                                                                                                    | regex      |
| 1 <code>mov eax, 0x10 ; a comment</code><br>2 <code>add rax, rbx</code>                                                                                                                                | x86        |

|   |                                               |            |
|---|-----------------------------------------------|------------|
| 1 | addi \$t0, \$zero, 10 # a comment             | mips       |
| 2 | sw \$t0, 0(\$sp)                              |            |
|   |                                               |            |
| 1 | if x goto L1                                  | tac        |
| 2 | param a                                       |            |
| 3 | call f, 1                                     |            |
|   |                                               |            |
| 1 | MOVE TEMP CONST 1 // a comment                | sirl       |
|   |                                               |            |
| 1 | ( <sub>1</sub> λx. x y) ( <sub>2</sub> λy. y) | lc         |
|   |                                               |            |
| 1 | Duck jens = new Duck(); # a comment           | rubberduck |

## 2.5 Full Reference

**TODO**