xlistings

https://github.com/EagleoutIce/xlistings

Florian Sihler*

Version v1.1.1 – 2026/11/11

1 Introduction 1
2 Package Options 2
3 Usage 3
4 Default Literates 5
5 Writing a Language 6
6 Full Reference 7
7 Origin 9
Index of commands and options 9

1 Introduction

The links in this manual are made with `code-link` *(see Section 1.2).*

This package extends the listings package with an easier front-end for code blocks of selected languages, number and segment highlighting, and non-selectable line numbers.[1] It is not compatible with the minted package, but provides a replacement for its interface. Compared to `listings`, it adds:

- Highlighting of numbers in code: `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`
- Highlighting of code segments, also across lines, with delimiters such as `|warn|`, `|err|`, and `|info|`: `int x = 0;`
- Support for umlauts and UTF-8 encoding (with the listingsutf8 package)
- Provides `autogobble` to remove leading spaces (with the lstautogobble 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:` (⟨), and `:to:` (→), see Section 4

*Released under the LPPL, version 1.3c or later.

[1] Whether a number is selectable depends on the viewer; images would be needed to prevent it.

- `\BadgeNextListing{<lang>}` to show a language tag at the start of the next listing (paragraph). `\xlstmintedwithlangbadge` and `\xlstmintedwithoutlangbadge` switch it on and off for all `fakeminted` environments.[2]

1.1 Loading the Package

Load the package with:

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

This document is typeset with `xlistings`. To get rid of the bordered look of the code blocks, use `\xlstsetstyle{plain number}` to switch the look and feel, or the `style` option:

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

For borderless blocks without line numbers, use `plain`:

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

1.2 Linking Code with `code-link`

The separate package `code-link` links the code in your listings to the place where it is defined or documented. You register a string where it is defined, and every occurrence of this string in a listing becomes a hyperlink. It works with `xlistings`, with `listings`, and with `minted`, and it does not need `xlistings`. It has a manual of its own, `code-link-doc.pdf`. This manual uses it: every command and option in its code links to where it is documented.

1.3 Dependencies

This package loads `kvoptions`, `etoolbox`, `xcolor`, `pgfkeys`, `lstautogobble`, and `listings`, and the following on demand:

- `needspace` for `guardspace`
- `tikz` for `highlights`
- `listingsutf8` and `accsupp`, if installed. Without `accsupp`, line numbers stay selectable; without `listingsutf8`, you lose the UTF-8 support.

All of these packages are part of common LaTeX distributions.

2 Package Options

`print` *package option*

Saves ink: code blocks get no background, and certain keywords are bold or italic so they stay readable in monochrome prints. See also `digital` for the complementary option.

`digital` *default, package option*

The opposite of `print`, and the default.

`guardspace` *default, package option*

Keeps the first lines of a listing together with its start, so a bordered block does not begin with a stray

[2] Redefine `\xlstlangbadgestyle` to restyle the badges; `accsupp` keeps them unselectable.

stripe at the bottom of a page. The number of lines is set with `\xlstguard`. On by default; `guardspace=false` disables it.

`numinpar` *package option*
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*
Highlights numbers in code blocks. On by default; `hlnumbers=false` disables it.

`upshape` *package option*
Inline commands normally follow the surrounding text and may be italic. This option forces an upright shape. For the font size, see `inlinesize`.

`inlinesize` *package option*
Inline commands normally adapt to the surrounding font size. This option fixes the size instead; set it with `\lstfs`. For the font shape, see `upshape`.

`highlights` *package option*
Loads tikz to highlight code segments with a color or a background (see the end of Section 5). Work in progress.

`fakeminted` *default, package option*
Provides 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. On by default; `fakeminted=false` disables it.

`extendednums` *package 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 a "d" behind a three. A language switches the letters off completely with `number suffixes={}`.

`debug` *package option*
Prints debug information to the console.

`style` *string, package option*
Selects the default style, like `\xlstsetstyle`. The default is `default`.

3 Usage

After loading the package as described in Section 1.1, you can load a language with `\LoadLanguages`:

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

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:

  ```latex
  \begin{java}
  ...
  \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 `\xlstc{<lang>}{<code>}`, `\xlstb{<lang>}{<code>}`, and `\xlsti{<lang>}{<file>}`, which take the language as an argument. This is useful to reach languages such as `x86` which contain numbers.
- 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`:

```latex
\xlstlangoverride{latex}{
    morekeywords=[5]{\\commandA, \\commandB},
    lineskip=42pt,
    % ...
}
```

As you can see, the sample above skips a given line number, which we achieve by using the following command: `\xlstblacklistlinenumbers{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).

In the `latex` language, a hyphen is part of a word, so `round-precision` can be a keyword. A command right after a hyphen is still highlighted on its own (`xlstcsbreak`):

```latex
\xlstlangoverride{latex}{morekeywords=[3]{round-precision}}
```

```latex
\sisetup{round-precision=2}\vspace{-\baselineskip}
```

The keys `alsoletter` and `alsoother` replace the characters of the language instead of adding to them, so repeat them when you change them. For `latex`, these are:

```latex
alsoletter={\\\#@*-}, alsoother={!,"_^}
```

If you want to add new literates for whatever reason, you can use the new listings key `add to literate`:

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

With this active, writing `sample` will result in π (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`:

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

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

```latex
% sample comment
```

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

```latex
\xlstDefineStyles{%
    {comments: \color{green}\bfseries},%
}
```

Now, comments will be displayed in green and bold:

```latex
% sample comment
```

This command is currently rather rigid and subject to change!

4 Default Literates

A *literate* replaces a character sequence by typeset material. The following ones are available in every listing, so you can write in plain text what listings cannot express, and they are the reason a few sequences, such as `:to:`, are unusable verbatim.

Input	Result	Meaning
`:ldots:`, `:cdots:`, `:cdot:`	. . ., ⋯, ·	ellipsis, centered ellipsis and dot
`:to:`, `:yields:`	→, →	arrows
`:lan:`, `:ran:`	⟨, ⟩	angle brackets
`:equiv:`, `:circ:`	≡, ∘	≡ and ∘
`:<:`, `:>:`	⟨, ⟩	gray angle brackets
`:lambda:`	λ	a lambda
`:br:`	a↩	a marker for a line break, not copied
`:bs:`	\	a backslash in the keyword color
`:bcmd:`, `:ecmd:`	\x	begin and end of a colored command
`:percent:`, `:!:`	%, !	characters that are special to your language
`:star:`, `:minus:`, `:ul:`	*, -, _	*, -, _
`:dollar:`, `:bmath:`, `:emath:`	$, $, $	a dollar in the number color
`:plaindollar:`	$	a plain dollar
`:lmath:`, `:rmath:`, `:ldmath:`, `:rdmath:`	\(, \), \[, \]	the math delimiters of LaTeX
`:float:`, `:exp:`	1f, 1e	suffix letters in the number color
`:space:`, `:ws:`	a b	an explicit space
`:c:`	int	nothing, but it ends the current token, so no keyword or number is recognized across it

Beyond these, the following are always available:

- Digits and decimal points, which are the basis of the `hlnumbers` option. With `extendednums`, the digit separator _ and the prefixes `0x`, `0b`, and `0o` join them.
- Accented letters, which `listings` cannot handle on its own: `x ä ö ü Ä Ö Ü ß á é í ó ú Á É à è ì ò ù â ê î ô û ñ Ñ ã õ ç Ç å Å æ Æ ø œ Œ € £ ¿ ¡`. An inline command must not *start* with such a letter, which is a limitation of `listings` itself; environments have no such restriction.
- Language specific literates, listed below, and the string escapes (`\n`, `\t`, ...) of the `C`-like languages.

The following literates belong to a single language:

Language	Input	Result	Meaning
`bash`	`:bashdollar:`	`$`	a dollar in the keyword color
	`:asyncamp:`	`&`	an ampersand that is no operator
`lc`	`:-alpha->:`, `:-eta->:`	$\xrightarrow{\alpha}$, $\xrightarrow{\eta}$	conversion arrows
	`:-beta->:`, `:-beta5->:`	$\xrightarrow{\beta}$, $\xrightarrow{\beta_5}$	reduction arrows
`tac`	`:_0:`, ..., `:_9:`	x_0, ..., x_9	a gray subscript digit
	`:__:`	x	an empty subscript
`mips`	`:hide$:`		nothing; it hides a dollar that would highlight a register

Further ones can be added with `add to literate`, as shown in Section 3.

5 Writing a Language

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

```latex
\lstdefinelanguage{lRubberduck}{
    comment=[l]{\#},
    morekeywords = {Quack, new},
    morekeywords = [2]{Duck}
}
\RegisterLanguage{rubberduck}{lRubberduck}
```

Keep the following in mind 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. Override it only if the language needs something else.
- The keyword classes `[1]` to `[6]` are styled as `keywordA`, `keywordB`, `keywordC`, `keywordD`, `literals`, and `command`. A language may 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 `\xlsthlwarning`, `\xlsthlerror`, and `\xlsthlinfo` are available as delimiter styles (they fall back to a plain color if the `highlights` option is not active), as are `\xlsthlbox{<color>}` and `\xlsthlboxd{<tikz keys>}`.

A segment may span several lines and may start anywhere on a line. The indentation in front of it stays outside the highlight, on its first line as well as on every further one:

```latex
\begin{ts}
class Point {
    |info|x = 0;|info|
    move(dx: number) {
        |warn|this.x = this.x
            + dx;|warn|
        |err|return y;|err|
    }
}
\end{ts}
```

```ts
class Point {
    x = 0;
    move(dx: number) {
        this.x = this.x
            + dx;
        return y;
    }
}
```

6 Full Reference

All commands below act locally, so wrap them in a group to limit their effect. Commands that contain an `@` in their name are internal and not part of the interface.

Loading and registering languages

`\LoadLanguages{<lang>,...}`
Loads the definitions of the given, comma-separated languages (`\LoadLanguage` is an alias). For each language, the file `xlistings-<lang>.cfg` is searched first in the current directory and then in `\xlstLanguageSearchPath`; the unprefixed `<lang>.cfg` is accepted as a fallback, so private definitions keep working. A language is loaded at most once, so definitions may require each other. If no definition is found, a warning is issued.

`\xlstLanguageSearchPath`
Comma-separated list of directories searched by `\LoadLanguages`; it defaults to `{langs/}`. The file name prefix (default `xlistings-`) is stored in `\xlstLanguageFilePrefix`.

`\RegisterLanguage[<aliases>]{<name>}{<lst-language>}`
Used by language files to register the language `<lst-language>` (a `listings` language) under `<name>`. This creates the environments, inline commands and (with `fakeminted`) the `minted` names described in the introduction. The optional, comma-separated aliases are additional `minted` names for the language.

`\xlstlangoverride[*]{<lang>}{<keys>}`
Appends the `listings` keys to the definition of the loaded language `<lang>`; the starred form replaces the previous overrides instead.

Styles and colors

`\xlstsetstyle{<style>}`
Selects the default style (`default`, `nonumber`, `plain`, or `plain number`), as the `style` option does.

`\lstcolorlet[<name>]{<color>}`
Sets the color of a highlighting class to an existing color. Valid names are `keywordA`, `keywordB`, `keywordC`, `numbers`, `numprefix`, `literals`, `comments`, `highlight`, `command`, and `linenumbers`.

`\lstcolordef[<mode>]{<name>}{<code>}`
Like `\lstcolorlet`, but defines the color from a color specification (the model `<mode>` defaults to `RGB`).

`\xlstDefineStyles{{<class>: <code>},...}`
Redefines the formatting of the given classes (`basic`, `keywordA`–`keywordD`, `numbers`, `linenumbers`, `literals`, `comments`, `highlight`, ...). `\xlstSetStyle{<class>}{<code>}` redefines a single one.

`\xlstGetStyle{<class>}`
Switches to the formatting of a class; `\xlstGet{<class>}{<text>}` typesets `<text>` in it within a group. `\xlstStyles` shows all styles at once.

Layout

`\xlstSetLeftMargin{<length>}`
Sets the left margin of listings. `\xlstSetRightMargin`, `\xlstSetNumSep` (distance of the line numbers), and `\xlstSetFrameRule` (frame width) work alike.

`\lstfs[*][<linespread>]{<size>}`
Sets the font size (in pt) and line spread of listings; the starred form takes the size of the surrounding text. `\xlstNumFs{<size command>}` sets the font size of line numbers only.

`\xlstguard{<lines>}`
Number of baseline skips that must remain on a page before a listing starts (default `.8`), if `guardspace` is active.

`\xlstPreBreak{<code>}`
Material shown at the end of a line that was broken in a listing; `\xlstInlinePreBreak` does the same for inline code.

`\xlstInlineKeys{<keys>}`
Keys appended to every inline listing after its style, so that they win over it, for example `breaklines=false,prebreak={}` to keep inline code from breaking.

`\xlstInlineLead{<material>}`
Inline code opens with a thin space, which is dropped at the start of a paragraph. When a wrapper such as `\href` encloses the inline code, wrap it in `\xlstInlineLead` so that the space sits outside the wrapper and vanishes at a line break as well.

`\xlstSetInputDefaults{<keys>}`
Default keys for the `\i<lang>` and `\xlsti` commands; `\xlstRemoveInputBackground` removes frame and background of included files.

Line numbers and numbers

`\xlstblacklistlinenumbers[*]{<numbers>}`
Hides the given, comma-separated line numbers. By default (`\xlstblacklistisghost`), hidden lines do not count, so later numbers close the gap. With the starred form, or after `\xlstblacklistispresent`, later lines keep their number.

`\xlstdisablenumhl`
Turns the highlighting of numbers off, `\xlstenablenumhl` turns it back on (see `hlnumbers`).

Listings keys

`add to literate` *listings key*
Appends to the `literate` list instead of replacing it, as shown in Section 3.

`number suffixes` *listings key*
Letters that may follow a number and still belong to it (default `eE`), if `extendednums` is active.

`language addons` *listings key*
Applies the keys given to `\xlstlangoverride` for the language. The environments and inline commands set it for you.

`xlstcsbreak` *listings key*
Starts a new word at each backslash (except within `\\`), so `-\baselineskip` highlights the command although the hyphen is a letter. Set by the `latex` language.

Code environments for `minted`

`\xlstmintedmap{<from>}{<to>}`
Lets the `minted` language `<from>` refer to the language `<to>`.

`\xlstsetmintedstyle{<style>}`
Selects the style of `minted` environments (default: the current `\xlstsetstyle`).

`\xlstmintedwithlangbadge`
Shows a language badge in all `minted` environments. `\xlstmintedwithoutlangbadge` turns it off again. With `fakeminted`, `\BadgeNextListing{<lang>}` adds a badge to the next listing only, and `\xlstlangbadgestyle{<lang>}` is the macro that styles it.

`\xlstc[<keys>]{<lang>}{<code>}`
The language-generic form of `\c<lang>`; `\xlstb` and `\xlsti` are the forms of `\b<lang>` and `\i<lang>`.

7 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.

Index of commands and options

:!: (literate), 5
:-alpha->: (literate), 6
:-beta->: (literate), 6
:-beta5->: (literate), 6
:-eta->: (literate), 6
:<: (literate), 5
:>: (literate), 5
:_0: (literate), 6
:_9: (literate), 6
:__: (literate), 6
:asyncamp: (literate), 6
:bashdollar: (literate), 6
:bcmd: (literate), 5
:bmath: (literate), 5
:br: (literate), 5
:bs: (literate), 5
:c: (literate), 5
:cdot: (literate), 5
:cdots: (literate), 5
:circ: (literate), 5
:dollar: (literate), 5
:ecmd: (literate), 5
:emath: (literate), 5
:equiv: (literate), 5
:exp: (literate), 5
:float: (literate), 5
:hide$: (literate), 6
:lambda: (literate), 5
:lan: (literate), 5
:ldmath: (literate), 5
:ldots: (literate), 5
:lmath: (literate), 5
:minus: (literate), 5
:percent: (literate), 5
:plaindollar: (literate), 5
:ran: (literate), 5
:rdmath: (literate), 5
:rmath: (literate), 5
:space: (literate), 5
:star: (literate), 5
:to: (literate), 5
:ul: (literate), 5
:ws: (literate), 5
:yields: (literate), 5
add to literate (listings key), 9
debug (option), 3
digital (option), 2
extendednums (option), 3
fakeminted (option), 3
guardspace (option), 2
highlights (option), 3
hlnumbers (option), 3
inlinesize (option), 3
language addons (listings key), 9
\LoadLanguages, 7
\lstcolordef, 8
\lstcolorlet, 7
\lstfs, 8
number suffixes (listings key), 9
numinpar (option), 2
print (option), 2
\RegisterLanguage, 7
style (option), 3
upshape (option), 3
\xlstblacklistlinenumbers, 8
\xlstc, 9
xlstcsbreak (listings key), 9
\xlstDefineStyles, 8
\xlstdisablenumhl, 8
\xlstGetStyle, 8
\xlstguard, 8
\xlstInlineKeys, 8
\xlstInlineLead, 8
\xlstlangoverride, 7
\xlstLanguageSearchPath, 7
\xlstmintedmap, 9
\xlstmintedwithlangbadge, 9
\xlstPreBreak, 8
\xlstSetInputDefaults, 8
\xlstSetLeftMargin, 8
\xlstsetmintedstyle, 9
\xlstsetstyle, 7