xlistings v0.1.0

opinionated `listings` extensions

Florian Sihler

October 26, 2025

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,[1] 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:

```
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 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:` (⟨), `: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.[2]

1.1 Loading the Package

To load the package, simply use:

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

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] 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.

[2] 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.

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

Or, using `\xlstsetstyle{plain}`:

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

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:

- `kvoptions`
- `needspace`(`guardspace`)
- `etoolbox`
- `xcolor`
- `pgfkeys`
- `lstautogobble`
- `listingsutf8`(*try*)
- `listings`
- `accsupp`(*try*)
- `tikz`(`highlights`)

"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 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. To disable this behavior, use `fakeminted=false`.

`extendednums` *package option*
: This option causes numbers with underscores or the classic `0x`-, `0b`- or `0o`-prefixes to be highlighted as well.

`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`:

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

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). 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 keys such as `keywordA`, `keywordB`, `keywordC`, `numbers`, `literals`, `comments`, `highlight`, and `command`:

```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}\scshape},%
}
```

Now, comments will be displayed in green and bold:

```latex
% sample comment
```

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 Full Reference

TODO