Czech Technical University in Prague
Faculty of Information Technology

tagsnip

Package Documentation

Author: Rostislav Brož
Year: 2026

tagsnip

tagsnip is a LaTeX package for including tagged code snippets from local files or remote URLs. It extracts marked sections of code and typesets them in a consistent style, supporting reproducible and maintainable documentation.

Installation

For parsing source files, *tagsnip* uses a Python package of the same name, which is published in the PyPI software repository [1]. In order for *tagsnip* to work, the backend has to be installed using the following command.

```
pip install tagsnip
```

Usage

tagsnip defines the command \IncludeCode, which is used as follows:

```
\IncludeCode[options]{source}{tag}{language}{caption}
```

The optional argument *options* allows the user to override code formatting settings. In the command below, the options *firstnumber*, *fontsize*, and *style* were used. They set line numbering to start from number 1, the font size to *scriptsize*, and the syntax highlighting style to *monokai*. The options must exist in the Minted package [2] and must be separated by commas. It is also important to mention that, without changing the *firstnumber* option, *tagsnip* numbers lines according to their original numbering in the source file.

```
\IncludeCode[firstnumber=1, fontsize=\scriptsize, style=monokai]
    {source}{tag}{language}{caption}
```

The mandatory argument *source* defines the source from which the code snippet should be taken. The argument can be either a path to a local file or a link to a web page. *tagsnip* can distinguish these possibilities automatically. The mandatory argument *tag* specifies a unique keyword that identifies one code snippet. The tag must be separated by exactly one space and must follow the phrases *tagsnip-start* and *tagsnip-end*, which delimit the snippet (see Snippet 1).

```
# tagsnip-start tag1
def main():
    x = 1
    y = 2
    print(x + y)

    return 0
# tagsnip-end tag1
```

Snippet 1: Example of tag usage.

With the mandatory argument *language*, the user selects the programming language whose syntax highlighting style should be used for the snippet. The last argument is the caption, which may be empty. However, the braces must still be written even for an empty caption.

Local snippet

To demonstrate how the \IncludeCode command works for local files, we use the file *example.py* from Snippet 1. In this file, the function *main()* is enclosed by the tag *tag1*. The command for displaying the function code in the document would therefore look as follows:

```
\IncludeCode{example.py}{tag1}{Python}{Úryvek 2: ...}
```

This command produces Snippet 2:

```
def main():
    x = 1
    y = 2
    print(x + y)

    return 0
```

Snippet 2: Snippet delimited by the tag *tag1* from Snippet 1.

Snippet from a remote URL

The home repository of *tagsnip* on GitHub also contains the file *example2.py*, which includes a function for extracting a snippet. Part of this function is enclosed by the tag *tag2*. We display it using the following command:

```
\IncludeCode{raw.githubusercontent.com/brozrost
    /tagsnip/main/docs/example2.py}{tag2}{Python}{Úryvek 3: ...}
```

The result of this command is Snippet 3:

```
    for i, line in enumerate(lines):
        if marker_matches(line, start_marker):
            if start_index is not None:
                raise TagsnipError(f"Multiple start tags found for
                    '{tag}'")

            start_index = i

    if start_index is None:
        raise TagsnipError(f"Start tag not found for '{tag}'")
```

Snippet 3: Code snippet from a remote URL.

Architecture

tagsnip consists of two main parts: a LaTeX frontend package and an external backend utility written in Python. The frontend defines the command \IncludeCode and handles the typesetting of the resulting snippets using the Minted package. The backend is responsible for accessing source files, finding marked sections, and extracting them into temporary files. During document compilation, the frontend uses shell escape to run the backend utility, passes it the path to the source file and the name of the tag, and then inserts the generated temporary file into the document.

Backend

The Python package *tagsnip* is a simple command-line interface that accepts a source file, a tag name, and a path to an output file. The interface loads the source file, finds the corresponding pair of *tagsnip-start* and *tagsnip-end* markers, and then extracts the text between them.

The package also checks error states. An error state means, for example, a non-existent source file, a missing tag, or multiple occurrences of the same tag in the source file. If an error is detected, the backend raises an exception, which interrupts document compilation and prints the corresponding error message. For working with local files, the package uses the *pathlib* module; for finding tags, it uses the *re* module; and for loading remote files over HTTP, it uses the *requests* library.

Limitations

- *tagsnip* requires the LuaLaTeX compiler.
- During document compilation, shell escape must be enabled using the *–shell-escape* option.
- The Python backend must be installed and available in the system PATH under the command *tagsnip*.
- The package depends on the Minted package.
- Remote source files must be available over HTTP(S) and in the format of a readable text file.

References

[1] Rostislav Brož. Home page of the tagsnip package. `https://pypi.org/project/tagsnip/`, 2026.

[2] Geoffrey M. Poore. The minted package: Highlighted source code in LaTeX. `https://tug.ctan.org/macros/latex/contrib/minted/minted.pdf`, 2026.