

## 0.1 minted

```
\usepackage{listings}
```

Minted uses Pygments for syntax highlighting.

Install Python and then Pygments.

```
1 $ pip install Pygments
```

To use Pygments on L<sup>A</sup>T<sub>E</sub>X, you need to pass `-shell-escape` flag to L<sup>A</sup>T<sub>E</sub>X.

```
1 $ lualatex -shell-escape <file>
```

If you want to compile L<sup>A</sup>T<sub>E</sub>X document containing minted with Visual Studio Code and LaTeX Workshop Plugin, add the following to `settings.json`.

```
1 {
2   "latex-workshop.latex.tools": [
3     {
4       "name": "lualatex",
5       "command": "lualatex",
6       "args": [
7         "-shell-escape",
8         "-synctex=1",
9         "-interaction=nonstopmode",
10        "-file-line-error",
11        "%DOC%"
12      ],
13      "env": {}
14    },
15    {
16      "name": "bibtex",
17      "command": "bibtex",
18      "args": [
19        "%DOCFILE%"
20      ],
21      "env": {}
22    }
23  ],
24  "latex-workshop.latex.recipes": [
25    {
26      "name": "lualatex",
27      "tools": [
```

```

28         "lualatex"
29     ]
30 },
31 {
32     "name": "lualatex -> bibtex -> lualatex * 2",
33     "tools": [
34         "lualatex",
35         "bibtex",
36         "lualatex",
37         "lualatex"
38     ]
39 }
40 ]
41 }

```

### 0.1.1 Inline code

`\mintinline{<language>}{<text>}`

### 0.1.2 Code block

For single line: `\mint{<language>}{<text>}`

```

1 \mint{python}{
2 print("Hello, world!")
3 }

```

```

1 print("Hello, world!")

```

For multiple lines: `\begin{minted} ... \end{minted}`

```

1 \begin{minted}{python}
2 def hello():
3     print("Hello, world!")
4 \end{minted}

```

```

1 def hello():
2     print("Hello, world!")

```

### 0.1.3 Input file

`\inputminted{<language>}{<file-path>}`

```

1 \inputminted{python}{hello.py}

```

```

1 def hello():
2     print("Hello, world!")

```

### 0.1.4 Captions and labels

Minted provides floating listing environment to use with caption and label.

```

1 \begin{listing}[H]
2   \mint{python}|print("Hello,
   ↳ world!")|
3   \caption{Code example}
4   \label{lst:example}
5 \end{listing}

```

```

1 print("Hello, world!")

```

Listing 1: Code example

## 0.1.5 Options

### Setting global minted options

inline & code blocks

```

1 \setminted{<options>}
2 \setminted[<language>]{<options>}

```

inline

```

1 \setmintedinline{<options>}
2 \setmintedinline[<language>]{<options>}

```

### Defining shortcuts

minted environment

```

1 \newminted{<language>}{<options>} % default environment-name:
   ↳ <language>code
2 \newminted[<environment-name>]{<language>}{<options>}
3
4 \begin{<environment-name>}
5 \end{<environment-name>}

```

mint command

```

1 \newmint{<language>}{<options>} % default macro-name: <language>
2 \newmint[<macro-name>]{<language>}{<options>}
3
4 \<macro-name>/<text>/ % ``/' can be replaces by any character

```

mintinline command

```

1 \newmintinline{<language>}{<options>} % default macro-name:
   ↳ <language>inline
2 \newmintinline[<macro-name>]{<language>}{<options>}

```

```

3
4 \<macro-name>/<text>/ % ``/' can be replaces by any character

```

inputminted command

```

1 \newmintedfile{<language>}{<options>} % default macro-name:
  ↳ <language>file
2 \newmintedfile[<macro-name>]{<language>}{<options>}
3
4 \<macro-name>{<file-path>}

```

### Available options

- autogobble (boolean): Remove gobble (leading whitespace)
- breaklines (boolean): Automatically break long lines
- frame (none | leftline | topline | bottomline | lines | single): Put lines around the code
- linenos (boolean): Linen numbers
- numbersep (dimension): Gap between numbers and start of line

```

1 \setminted{
2   autogobble,
3   breaklines,
4   frame=single,
5   linenos,
6   numbersep=2mm,
7 }

```