I. Use cases in LaTeX

Documenting a package or class is best done through use cases showing both the code and the corresponding result. [1]

Caution.

Version 3 of `minted` *cannot be used at the moment, as it contains bugs: see* `https://github.com/gpoore/minted/issues/401`*. We therefore force the use of version 2 of minted.*

1. "*Inline*" codes

The `\tdocinlatex` macro [2] can be used to type inline code in a similar way to `\verb` or like a standard macro (see brace management in the last case below). Here are some examples.

```
1: \tdocinlatex|$a^b = c$|                    \\
2: \tdocinlatex+\tdocinlatex|$a^b = c$|+ \\
3: \tdocinlatex{\tdocinlatex{$a^b = c$}}
```

1: $a^b = c$
2: \tdocinlatex|$a^b = c$|
3: \tdocinlatex{$a^b = c$}

Note.

The `\tdocinlatex` *macro can be used in a footnote: see below.* [a] *In addition, a background color is deliberately used to subtly highlight the codes* `\LaTeX` *.*

[a] `$minted = TOP$` has been typed `\tdocinlatex+$minted = TOP$+` in this footnote...

2. Directly typed codes

Example I.1 (Side by side)**.** *Using* `\begin{tdoclatex}[sbs] ... \end{tdoclatex}`*, we can display a code and its rendering side by side. Consider the following code.*

```
\begin{tdoclatex}[sbs]
    $A = B + C$
\end{tdoclatex}
```

This will produce the following.

```
$A = B + C$
```

$A = B + C$

Example I.2 (Following)**.** `\begin{tdoclatex} ... \end{tdoclatex}` *produces the following result, which corresponds to the default option* `std` *.* [3]

```
$A = B + C$
```

$A = B + C$

Example I.3 (Just the code)**.** *Via* `\begin{tdoclatex}[code] ... \end{tdoclatex}`*, we'll just get the code as shown below.*

```
$A = B + C$
```

[1] Code is formatted using the `minted` package.
[2] The name of the macro `\tdocinlatex` comes from "*in-line* LaTeX".
[3] `std` refers to the "*standard*" behaviour of `tcolorbox` in relation to the `minted` library.

Warning.

With default formatting, if the code begins with an opening bracket, the default option must be explicitly indicated. Consider the following code.

```
\begin{tdoclatex}[std]
    [Strange... Or not!]
\end{tdoclatex}
```

This will produce the following.

```
[Strange... Or not!]
```

[Strange... Or not!]

Another method is to use the `\string` *primitive. Consider the following code.*

```
\begin{tdoclatex}
    \string[Strange... Or not!]
\end{tdoclatex}
```

This will produce the following.

```
[Strange... Or not!]
```

[Strange... Or not!]