

I. Use cases in L^AT_EX

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

1. “Inline” codes

The `\tdoclatexin` macro² 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: <code>\tdoclatexin $a^b = c$ </code> <code>\\</code>	1: $a^b = c$
2: <code>\tdoclatexin+\tdoclatexin $a^b = c$ +</code> <code>\\</code>	2: <code>\tdoclatexin $a^b = c$ </code>
3: <code>\tdoclatexin{\tdoclatexin{$a^b = c$}}</code>	3: <code>\tdoclatexin{$a^b = c$}</code>

Note.

The `\tdoclatexin` 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 `\tdoclatexin+$minted = TOP$+` 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.

$A = B + C$

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.

¹Code is formatted using the `minted` package.

²The name of the macro `\tdoclatexin` comes from “*in-line* L^AT_EX”.

³`std` refers to the “*standard*” behaviour of `tcolorbox` in relation to the `minted` library.

```
[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!]
```