

I. Use cases in L^AT_EX

Documenting a package or a class is done efficiently using use cases showing both the code and the corresponding result.

1. “*Inline*” codes

The `\tdocinlatex` macro¹ can be used to type inline code in a similar way to `\verb`. Here are some examples.

1: <code>\tdocinlatex \$a^b = c\$ </code>	1: <code>\$a^b = c\$</code>
2: <code>\tdocinlatex+\tdocinlatex \$a^b = c\$ +</code>	2: <code>\tdocinlatex \$a^b = c\$ </code>

Note. The `\tdocinlatex` macro can be used in a footnote: see the bottom of this page².

2. Directly typed codes

Example 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.

<code>\$A = B + C\$</code>	$A = B + C$
----------------------------	-------------

Example 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 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.

[Strange... Or not!]

[Strange... Or not!]

¹The name of the macro `\tdocinlatex` comes from “*in-line L^AT_EX*”.

²`$minted = TOP$` was typed `\tdocinlatex+$minted = TOP$+` in this footnote.

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