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]

1 "*Inline*" codes

Example I.1 (Standard usage)**.** *The* `\tdoclatexin` *macro*[2] *can be used to type code in line in a similar way to* `\verb`*, or as a standard macro (see the handling of braces in the latter case below). Here are some examples of use.*[3]

```
<\tdoctcb{sbs}>
1: \tdoclatexin|$a^b = c$|                  ||
2: \tdoclatexin+\tdoclatexin|$a^b = c$|+ ||
3: \tdoclatexin{\tdoclatexin{$a^b = c$}}
```

<sbs> 1: `$a^b = c$`
2: `\tdoclatexin|$a^b = c$|`
3: `\tdoclatexin{$a^b = c$}`

Example I.2 (Possible options)**.** *As the* `\tdoclatexin` *macro is based on* `minted`*, you can use all the options taken into account by* `minted`*. Here are some examples.*

```
<\tdoctcb{sbs}>
1: \tdoclatexin{$a^b = c$}                ||
2: \tdoclatexin[style = bw]{$a^b = c$} ||
3: \tdoclatexin[style = igor,
                showspaces]{$a^b = c$}
```

<sbs> 1: `$a^b = c$`
2: `$a^b = c$`
3: `$a^b␣=␣c$`

Note.

The `\tdoclatexin` *macro can be used in a footnote: see below.*[a]

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

2 Directly typed codes

Example I.3 (Face to face)**.** *Displaying a code and its rendering side by side is done as follows where the macro* `\tdoctcb` *allows you to just type* `tdoctcb{sbs}` *instead of* `listing side text` *(*`sbs` *is for "***s·ide b·y s·ide***", while* `tcb` *is the standard abbreviation for* `tcolorbox`*). Note the use of rafters, not square brackets (more on this later).*

```
\begin{tdoclatex}<\tdoctcb{sbs}>
$A = B + C$
\end{tdoclatex}
```

This gives :

[1] Code is formatted using the `minted` and `tcolorbox` packages.
[2] The name of the macro `\tdoclatexin` comes from "***in·line*** LaTeX".
[3] A background colour is deliberately used to subtly highlight the `\LaTeX` codes.

Start of the real output

```
<\tdoctcb{sbs}>
$A = B + C$
```

$<sbs>\ A = B + C$

End of the real output

Example I.4 (Following). *`\begin{tdoclatex}...\end{tdoclatex}` produces the following result (this default setting is also obtained by using `\tdoctcb{std}`).*[4]

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

$A = B + C$

Example I.5 (Just the code). *Via `\tdoctcb{code}`, we'll just get the code as below.*

```
<\tdoctcb{code}>
$A = B + C$
```

$<code>\ A = B + C$

Example I.6 (Customise). *The `tdoclatex` environment accepts two types of optional argument.*

1. *Between classic square brackets, you can use any option taken into account by `minted`.*
2. *Between rafters, you can use any option taken into account by the environments obtained via `tcolorbox`.*

For example, the following modifications can be made if required.[5]

```
\begin{tdoclatex}%
    [linenos, style = igor]%
    <\tdoctcb{sbs},
     attach boxed title to top left = {yshift = -9pt},
     fonttitle                      = \bfseries,
     title                          = Local modifications,
     top                            = 10pt>
% Sometimes useful, but don't overuse it!
$A = B + C$
% End of this demonstration.
\end{tdoclatex}
```

This gives :

Start of the real output

```
%
    <\tdoctcb{sbs},
     attach boxed title to top left = {yshift = -9pt},
     fonttitle                      = \bfseries,
     title                          = Local modifications,
     top                            = 10pt>
% Sometimes useful, but don't overuse it!
$A = B + C$
% End of this demonstration.
```

$<sbs,\ attach\ boxed\ title\ to\ top\ left = yshift = -9pt,\ fonttitle = \bfseries,\ title = Local\ modifications,\ top = 10pt>\ A = B + C$

End of the real output

[4] `std` refers to the "*standard*" behaviour of `tcolorbox` in relation to the `minted` library.

[5] This documentation uses the options between rafters to obtain correct rendering of code producing shaded frames: see the section ?? on page ??.

Warning.

To obtain the default formatting for a code beginning with a bracket or a rafter, you'll need to do a bit of fiddling, as shown below.

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

This gives :

Start of the real output

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

[Strange... Or not!]

OR.

```
<>
\string<Strange... Or not!>
```

<> <Strange... Or not!>

End of the real output

Another method is to use the `\string` *primitive, as shown below.*

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

This gives :

Start of the real output

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

[Strange... Or not!]

OR.

```
<Strange... Or not!>
```

<Strange... Or not!>

End of the real output