

# I Presenting computer code

Some packages offer functions that require to code a little in Lua.<sup>1</sup> For these projects, the documentation must be able to present lines of code; this is why `tutodoc` makes it easy to do this, and much more.<sup>2</sup>

## Important.

*The tools in this section can also be used to present L<sup>A</sup>T<sub>E</sub>X code, but they should not be used for simple use cases, as the macros and environments presented next are for studying code, not just for using it: see the section ?? on page ?? to use the right tools for formatting L<sup>A</sup>T<sub>E</sub>X use cases.*

## 1 “Inline” codes

The `\tdoccodein` macro expects two arguments: the 1<sup>st</sup> indicates the programming language, and the 2<sup>nd</sup> gives the code to be formatted. It is possible to use an option identical to that proposed by `\tdoclatexin`: see the section ?? on page ?? . Here are some possible use cases.<sup>3</sup>

```
1: \tdoccodein{py}{print("OK" if i = 0 else "KO")} \\
2: \tdoccodein[style = bw]{py}{print("OK" if i = 0 else "KO")} \\
3: \tdoccodein[style = igor, showspaces]%
   {py}{print("OK" if i = 0 else "KO")}
-----
1: pyprint("OK" if i = 0 else "KO")
2: [style = bw]pyprint("OK" if i = 0 else "KO")
3: [style = igor, showspaces]pyprint("OK" if i = 0 else "KO")
```

## Note.

*The <https://pygments.org/languages/> page contains a complete list of supported languages with their short names. For example, it is possible to format *Brainfuck* like this sequence `bf+++++++>+++++++>+++++++>+++>«-]>+>+.+++++++..++>` which is used to display text *Hello*.*

## 2 Codes typed directly

Code can be typed directly into a document via `\begin{tdoccode} ... \end{tdoccode}` which expects an argument indicating the programming language, and any options between parenthesis and/or square brackets identical to those offered by `\begin{tdoclatex} ... \end{tdoclatex}`: see the section ?? on page ?? . In the following examples, the L<sup>A</sup>T<sub>E</sub>X codes for `tutodoc` are displayed in verbatim mode because the coloring of L<sup>A</sup>T<sub>E</sub>X codes is not correct.

**Example I.1** (Standard Operation).

```
\begin{tdoccode}{pl}
print "Who are you? ";
my $name = <STDIN>;

chomp($name);

if ($name eq "") {
    print "Ah, not very chatty today!";
} else {
    print "Hello $name";
    print "Amazing! Actually, not at all...";
}
\end{tdoccode}
```

<sup>1</sup>For mathematics, these include `luacas` and `tkz-elements`.

<sup>2</sup>As code formatting is done via the packages `minted` and `tcolorbox`, the macros and environments presented in this section allow code to be formatted in all the languages supported by `Pygments`, a Python project used behind the scenes by `minted`.

<sup>3</sup>A background color is deliberately used to subtly highlight the formatted codes. For example, typing `\tdoccodein{py}{funny = "ah"*3}` will produce `pyfunny = "ah"*3`.

```
pl print "Who are you? "; my name =< STDIN >;
chomp(name);
if (nameeq"")print" Ah, notverychattytoday!";
else print "Hello name";print" Amazing!Actually, notatall...";
```

**Example I.2** (One-off rendering customisation).

```
\begin{tdoccode}[style = solarized-light, linenos]%
    <leftrule = 22pt, colback = orange!5, colframe = red!35%>
    {lua}
io.write("Who are you?")
local name = io.read()

if name == "" then
    print("Ah, not very chatty today!")

else
    print("Hello “ .. name .. ”.")
    print("Amazing! Actually, not at all...")
end
\end{tdoccode}
```

```
[style = solarized-light, linenos]<leftrule = 22pt, colback = orange!5, colframe = red!35lua io.write("Who
are you?") local name = io.read()
if name == "" then print("Ah, not very chatty today!")
else print("Hello “ .. name .. ”.") print("Amazing! Actually, not at all...") end
```