I. Presenting computer code

Some packages offer functions that require to code a little in `Lua`.[1] 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.[2]

Important.

The tools in this section can also be used to present LaTeXcode, 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 LaTeX use cases.

1. "*Inline*" codes

The `\tdoccodein`[3] macro expects two arguments: the 1st indicates the programming language, and the 2nd 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.[4]

```
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: print("OK" if i = 0 else "KO")
2: print("OK" if i = 0 else "KO")
3: print("OK"␣if␣i␣=␣0␣else␣"KO")
```

Note.

The page `https://pygments.org/languages/` *contains a complete list of supported languages with their short names. For example, it is possible to format* `Brainfuck` *code like this sequence* `++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.` *which displays* `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 proposed by `\begin{tdoclatex}...\end{tdoclatex}`: see the section ?? on page ??.[5]

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

```
\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}
```

This gives :

[1] For mathematics, these include `luacas` and `tkz-elements`.

[2] 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`.

[3] The name of the macro `\tdoccodein` comes from "***in**-line **code***".

[4] A background color is used to subtly highlight the formatted codes. For example, typing `\tdoccodein{py}{funny = "ah"*3}` will produce `funny = "ah"*3`.

[5] Note that the coloring of the LaTeX codes is lexically correct, but semantically wrong.

```
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...";
}
```

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

```
\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}
```

This gives :

```
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
```