I. Présenter du code informatique

Certains packages proposent des outils utilisables via le langage `Lua` depuis un document LaTeX.[1] Pour ce type de projet, il est utile de pouvoir présenter des lignes de code `Lua`. Pour cette raison, `tutodoc` permet de faire cela aisément, et bien plus.[2]

Mise en garde.

Les outils de cette section permettent aussi de présenter du code LaTeX, mais il ne faut pas les utiliser pour de simples cas d'utilisation, car les macros et les environnements présentées juste après servent à étudier du code, et non juste à l'employer : se reporter à la section ?? page ?? pour faire appel aux bons outils pour la mise en forme de cas d'utilisation LaTeX.

1. Codes « *en ligne* »

La macro `\tdocodein` attend deux arguments : le 1er indique le langage de programmation, et le 2e donne le code à mettre en forme. Il est possible d'utiliser une option de fonctionnement identique à ce que propose `\tdoclatexin` : voir la section ?? page ??. Voici des cas d'utilisation possibles..[3]

```
1: \tdocodein{py}{print("OK" if i=0 else "KO")}              \\
2: \tdocodein[style = bw]{py}{print("OK" if i=0 else "KO")} \\
3: \tdocodein[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")
```

2. Codes tapés directement

XXX

```
\begin{tdocode}[style=solarized-light, linenos]%
               <top = 1cm, leftrule=5pt>%
               {lua}
io.write("Qui êtes-vous ? ")
local name = io.read()

if name == "" then
    print("Ah, pas très bavard aujourd'hui !")

else
    print("Bonjour " .. name .. ".")
    print("Épatant ! En fait, pas du tout...")
end
\end{tdocode}
```

```lua
io.write("Qui êtes-vous ? ")
local name = io.read()

if name == "" then
    print("Ah, pas très bavard aujourd'hui !")

else
    print("Bonjour " .. name .. ".")
    print("Épatant ! En fait, pas du tout...")
end
```

1. Pour les mathématiques, on peut citer `luacas` et `tkz-elements`.
2. La mise en forme des codes étant faite via les packages `minted` et `tcolorbox`, les macros et les environnements présentés dans cette section permettent la mise en forme de codes dans tous les langages supportés par `Pygments`, un projet `Python` utilisé en coulisse par `minted`.
3. Une couleur de fond est volontairement utilisée pour subtilement faire ressortir les codes `mis.en.forme`.