DM n°4 - FDI

 $Hugo \ Salou$



5 décembre 2024

Dans la première solution, on n'utilise pas de quine mais on instaure une propriété très utilise : « pour toute lettre du programme, elle apparait exactement 10 fois ». Pour réaliser cela, on cache beaucoup de caractères dans un commentaire (comme par exemple f ou [).

```
import sys; print(10 if sys.argv[1] in "\"\\n#ev
  \hookrightarrow .famit<sub>\(\sigma\)</sub>;p(:n0[rl)1s]gyo" else 0) #
  \hookrightarrow eeeeeeevvvvvvvv.....
  \hookrightarrow fffffffaaaaaaaaammmmmmmiiiitttttttppppppp
  \hookrightarrow 1111111ssss]]]]]]]]ggggggggyyyyyyooooooo
#
#
#
#
#
#
#
#
#
```

Code 1 | Solution sans quine au DM 4

C'est moche, pas toujours généralisable, mais ça répond au DM.

J'ai quand même proposé une seconde solution à ce DM n'utilisant pas cette technique mais utilisant une quine, c'est à dire un programme qui s'écrit lui même. La solution est encore plus concise mais utilise des « *string format* » (à la printf en C).

```
 \begin{array}{l} x = \text{"x=\%c\%s\%c; y=x\%c(chr(34),x,chr(34),chr(37),chr} \\ \hookrightarrow & (10)); import \  \  \, sys; print(y.count(sys.argv[1])) \\ \hookrightarrow & )\%c"; y = x\%(chr(34),x,chr(34),chr(37),chr(10)) \\ \hookrightarrow & ); import \  \, sys; print(y.count(sys.argv[1])) \\ \hline & Code 2 \mid Solution \  \, avec \  \, quine \  \, au \  \, DM \  \, 4 \\ \end{array}
```

L'idée est simple, dans ${\tt x}$, on écrit le code source. Il contient trois parties, le début de la définition de ${\tt x}$ (i.e. « ${\tt x}$ = »), la chaîne ${\tt x}$

elle-même (où l'on utilise ici le « \$s » pour interpoler la valeur de x dedans), et enfin le reste du code (la partie qui compte le nombre de caractères, et qui l'affiche en console). Les caractères problématiques, comme les guillemets, les retours à la ligne (parce que le backslash), le symbole pour-cent, etc, sont interpolés aussi dans la chaîne avec « \$c » et la fonction ord qui prend un entier et renvoie le caractère ASCII à l'indice donné dans la table.