Exercice 231, page 284

```
def xor(x, y):
    return (x and not y) or (not x and y)
```

Exercice 232, page 284 On note $f_1(x,y,z) = (x \wedge y) \vee (\neg y \wedge z)$ et $f_2(x,y,z) = (x \vee \neg y) \wedge (y \vee z)$.

Une première façon de montrer l'égalité $f_1(x,y,z) = f_2(x,y,z)$ est de vérifier que les tables de vérité de ces fonctions sont les mêmes.

x	y	z	$f_1(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x	y	z	$f_1(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

La deuxième méthode consiste à utiliser les propriétés des opérateurs logiques pour transformer par exemple la deuxième expression en la première.

$$\begin{aligned}
(x \vee \neg y) \wedge (y \vee z) &= ((x \vee \neg y) \wedge y) \vee ((x \vee \neg y) \wedge z) \\
&= (x \wedge y) \vee (y \wedge \neg y) \vee (x \wedge z) \vee (\neg y \wedge z) \\
&= (x \wedge y) \vee 0 \vee (x \wedge z) \vee (\neg y \wedge z) \\
&= (x \wedge y) \vee (x \wedge z) \vee (\neg y \wedge z) \\
&= (x \wedge y) \vee ((x \wedge z) \wedge (y \vee \neg y)) \vee (\neg y \wedge z) \\
&= (x \wedge y) \vee (x \wedge z \wedge y) \vee (x \wedge z \wedge \neg y) \vee (\neg y \wedge z) \\
&= ((x \wedge y) \wedge (1 \vee z)) \vee ((\neg y \wedge z) \wedge (1 \vee x)) \\
&= ((x \wedge y) \wedge 1) \vee ((\neg y \wedge z) \wedge 1) \\
&= (x \wedge y) \vee (\neg y \wedge z)
\end{aligned}$$

Exercice 233, page 284 Il y a plein de solutions. En voici deux :

$$\begin{aligned}
f(x,y) &= \neg(x \oplus y) \\
&= (x \wedge y) \vee (\neg x \wedge \neg y)
\end{aligned}$$

La table de vérité est la suivante :

x	y	$f(x,y)$
0	0	1
0	1	0
1	0	0

Exercice 234, page 284 La table est la suivante :

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Cette fonction détermine si exactement une variable vaut 1. Une expression plus simple est

$$f(x,y,z) = (\neg x \wedge (y \oplus z)) \vee (x \wedge \neg y \wedge \neg z).$$

Exercice 235, page 285 Le premier additionneur 1 *bit* se charge d'additionner les *bits* de poids faible. Son entrée c_0 vaut 0. Sa sortie c (la retenue) est envoyée sur l'entrée c_0 du second additionneur 1 *bit*, qui se charge de l'addition des *bits* de poids fort.

Supposons que le premier nombre soit e_0e_1 et le second e_2e_3. Alors, la table de vérité est la suivante, où s_0s_1 est le résultat et c la retenue sortante.

entrées				sorties		
e_0	e_1	e_2	e_3	s_0	s_1	c
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	1	1	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	1	1