Exercice 1 : Soit P et Q deux propositions logiques. Déterminer la table de vérité de la proposition « non (P) ou Q ».

P	Q	non (P) ou Q
0	0	
0	1	
1	0	
1	1	

Tableau 1 – non (P) ou Q

Exercice 2 : Soit P et Q deux propositions logiques. On considère une proposition T(P,Q), construite à partir des propositions P et Q, dont la table de vérité est donnée ci-dessous.

P	Q	T(P,Q)
F	F	F
F	V	F
V	F	V
V	V	F

Tableau 2 – T(P,Q)

Parmi les propositions suivantes, laquelle est logiquement équivalente à T(P,Q) ?

— Q et non(P)

— non(Q) ou P

— non(Q) et P

— Q ou non(P)

Exercice 3 : Soit A, B et C trois propositions logiques. Déterminer la table de vérité de la proposition « C et (A ou B) ».

A	B	C	C et (A ou B)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tableau 3 – C et (A ou B)

Exercice 4 : Montrer l'égalité suivante :

$$(x \wedge y) \vee (\neg y \wedge z) = (x \vee \neg y) \wedge (y \vee z)$$

Christophe Viroulaud

Exercice 5 : On considère la fonction booléenne suivante :

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

1. Donner sa table de vérité.
2. Que fait cette fonction ? Dans quel cas la sortie vaut 1 ?

Exercice 6 : En combinant 2 additionneurs *1 bit* nous pouvons obtenir un additionneur *2 bits*. Notons e_0e_1 et e_2e_3 deux nombres binaires.
Le premier additionneur se charge d'additionner les bits de poids faible.

1. Quels bits additionnent le premier additionneur ?
2. Que vaut l'entrée c_0 de cet additionneur ?
3. Où est envoyé la sortie c de ce premier additionneur ?
4. Combien de lignes y-aura-t-il dans la table de vérité de l'additionneur 2 bits ?
5. Compléter la table de vérité de l'additionneur 2 bits ci-après :

e_0	e_1	e_2	e_4	s_0	s_1	c
0	0	0	0	0	0	0
...						

Tableau 4 – Additionneur 2 bits

Exercice 7 :

1. Que va afficher le programme suivant si x vaut :

   10 100 -1 0 110

```
if x >= 0 and x < 100:
    print("salut")
else:
    print("bonjour")
```

2. Que va afficher le programme suivant si x et y valent :

x	0	-1	0	110
y	-1	0	10	-1

```
if x >= 0 or y < 0:
    print("salut")
else:
    print("bonjour")
```

3. La fonction **xor** n'existe pas nativement en Python. Construire les tables de vérité de :
   — $x \wedge \neg y$
   — $\neg x \wedge y$

   En déduire une fonction Python **xor(x: bool, y: bool) → bool** qui simule le OU EXCLUSIF.

Christophe Viroulaud