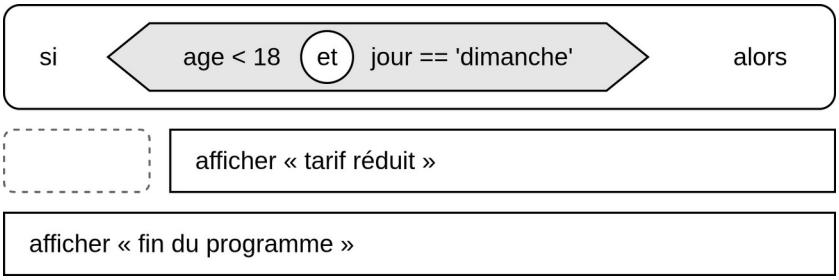
Nom :	Prénom :	Classe :

NSI 1re — Opérateurs logiques

Éléments du programme : Représentation des données - types et valeurs de base → Valeurs booléennes (0 = False, 1 = True). Opérateurs booléens (and, or, not, xor). Table d'une expression booléenne.

Opérateurs logiques

Un **opérateur logique** sert à relier plusieurs affirmations entre elles. Prenons un exemple :



← ici l'affirmation est composée des deux "sous-affirmations" `age < 18` mais aussi `jour == 'dimanche'`.

Ces deux "sous-affirmations" sont liées entre elles par l'opérateur logique `et`.

L'exemple schématisé ci-dessus peut être transcrit en Python de la façon suivante :

```
1  if age < 18 and jour == 'dimanche':
2      print('tarif réduit')
3  print('fin du programme')
```

Voici une explication du programme, ligne par ligne :

Ligne	Explication
1	Condition dans laquelle **l'affirmation** est : - La valeur de la variable `age` est inférieure à la valeur `18` ET - La valeur de la variable `jour` vaut `dimanche`
2	Instruction (bloc de code) qui sera exécutée, si **l'affirmation est vraie**.
3	Cette instruction ne fait pas partie de la condition. Elle sera donc exécutée *quoi qu'il arrive*.

Liste des opérateurs logiques

Opérateur logique	Description
AND	L'affirmation est vraie si toutes les affirmations liées sont vraies.
OR	L'affirmation est vraie si au moins une des affirmations liées est vraie.
NOT	L'affirmation est vraie si elle est fausse, et vice versa.
XOR	L'affirmation est vraie si une seule des affirmations liées est vraie.

Opérateur logique AND (et)

Avec l'opérateur logique "et" (« and », en anglais),
une affirmation est **vraie**, si **toutes les affirmations liées par cet opérateur sont vraies**.

Exemple en *langue humaine* :

10 est plus grand que 5	et	50 est plus petit que 100
Affirmation 1	*Opérateur*	*Affirmation 2*

Dans cet exemple, les deux affirmations liées sont vraies,
donc l'affirmation finale est vraie.

Une transcription possible en Python de cet exemple pourrait être :

```
>>> 10 > 5 and 50 < 100
True
```

Comme nous pouvons le voir, en Python l'opérateur "et" s'écrit `and`.

Mais que se passe-t-il si la première affirmation est vraie, mais que la seconde est fausse ? Ou inversement ? Heureusement, il est possible de lister toutes les combinaisons réalisables avec un opérateur logique dans un tableau :

Affirmation A	Affirmation B	A and B
Faux	Faux	Faux
Faux	Vrai	Faux
Vrai	Faux	Faux
Vrai	Vrai	Vrai

← Ce tableau est appelé *table de vérité*. Un concept développé par George Boole au XIX[e] siècle.

Chaque opérateur logique dispose ainsi de **sa propre table** de vérité[1].

➤ Exercice 1

Analysez les affirmations suivantes, puis déterminez leurs valeurs à Vrai ou Faux :

Affirmation	Valeur (à compléter)		Affirmation	Valeur (à compléter)
`True and False`			`1 >= 1 and 1 > 1`	
`False and True`			`0 > 2 and 2 > 1`	
`False and False`			`-1 < 0 and 1 > 0`	
`True and True`			`True and True and False`	
`1 > 2 and 2 > 1`			`1 > 0 and 0 < 2 and 1 == 1`	

1 Ces tables sont couramment utilisées en mathématiques (logique propositionnelle), en électronique (porte logique) et en informatique (tests).

Opérateur logique OR (ou)

Avec l'opérateur logique "ou" (« or », en anglais),
une affirmation est **vraie**, si **au moins une des affirmations liées par cet opérateur est vraie**.

Exemple en *langue humaine* :

10 est plus petit que 5	ou	50 est plus petit que 100
Affirmation 1	*Opérateur*	*Affirmation 2*

Dans cet exemple, l'une des deux affirmations liées est vraie,
donc l'affirmation finale est vraie.

Une transcription possible en Python de cet exemple pourrait être :

```
>>> 10 < 5 or 50 < 100
True
```

Comme nous pouvons le voir, en Python l'opérateur "ou" s'écrit `or`.

Voici la table de vérité de l'opérateur logique "ou" :

Affirmation A	Affirmation B	A or B
Faux	Faux	Faux
Faux	Vrai	Vrai
Vrai	Faux	Vrai
Vrai	Vrai	Vrai

➤ Exercice 2

Analysez les affirmations suivantes, puis déterminez leurs valeurs à Vrai ou Faux :

Affirmation	Valeur (à compléter)		Affirmation	Valeur (à compléter)
`False or False`			`1 >= 1 and 1 > 1`	
`True or True`			`1 > 0 or 1 == 2`	
`False or True`			`True or 1 > 0`	
`True or False`			`True or False or False`	
`0 < 1 or 1 > 0`			`1 < 0 or 0 > 1 or 1 > 0`	

➤ Exercice 3

Est-ce que le bloc de code sera exécuté après la condition Python `if 1 > 0 or 0 < 1:` ? .....

Opérateur logique NOT (« n'est pas »)

Avec l'opérateur logique "n'est pas" — ou opérateur "non", ou opérateur "de négation" (« not », en anglais), une affirmation devient **vraie**, si elle est fausse. Et une affirmation devient fausse, si elle est vraie.

Exemple en *langue humaine* :

négation	5 est plus petit que 10
Opérateur	*Affirmation*

Dans cet exemple, l'affirmation est vraie,
donc l'affirmation finale est fausse.

Une transcription possible en Python de cet exemple pourrait être :

```
>>> not 5 < 10
False
```

Comme nous pouvons le voir, en Python l'opérateur "de négation" s'écrit `not`.

Voici la table de vérité de l'opérateur logique "not" :

Affirmation A	not A
Faux	Vrai
Vrai	Faux

➤ Exercice 4

Analysez les affirmations suivantes, puis déterminez leurs valeurs à Vrai ou Faux.

Bien penser à **résoudre d'abord** l'affirmation **entre les parenthèses**.

Affirmation	Valeur (à compléter)		Affirmation	Valeur (à compléter)
`not False`			`not( False or False )`	
`not True`			`not( 0 > 2 and 2 > 1 )`	
`not( True or False )`			`not( 1 > 2 and 2 > 1 )`	
`not( True or True )`			`not( 1 >= 1 and 1 > 1 )`	
`not( False or True )`			`True and not False`	

Note : l'opérateur logique NOT a une **priorité plus haute** que les autres opérateurs.

Opérateur logique XOR (ou exclusif)

Avec l'opérateur logique "ou exclusif" (« xor », en anglais),
une affirmation est **vraie**, si **uniquement <u>une et une seule</u> des affirmations liées par cet opérateur est vraie**.

Exemple en *langue humaine* :

10 < 5	ou exclusif	50 < 100	ou exclusif	2 > 1
Affirmation 1	*Opérateur*	*Affirmation 2*	*Opérateur*	*Affirmation 3*

Dans cet exemple, deux des trois affirmations liées sont vraies,
donc l'affirmation finale est fausse.

Une transcription possible en Python de cet exemple pourrait être :

```
>>> (10 < 5) ^ (50 < 100) ^ (2 > 1)
False
```

Comme nous pouvons le voir, en Python l'opérateur "ou exclusif" s'écrit `xor`.

Voici la table de vérité de l'opérateur logique "ou exclusif" :

Affirmation A	Affirmation B	A xor B
Faux	Faux	Faux
Faux	Vrai	Vrai
Vrai	Faux	Vrai
Vrai	Vrai	Faux

➤ Exercice 5

Analysez les affirmations suivantes, puis déterminez leurs valeurs à Vrai ou Faux :

Affirmation	Valeur (à compléter)		Affirmation	Valeur (à compléter)
`False ^ False`			`(1 >= 1) ^ (1 > 1)`	
`True ^ True`			`(1 > 0) ^ (1 == 2)`	
`False ^ True`			`True ^ (1 > 0)`	
`True ^ False`			`True ^ False ^ False`	
`(0 < 1) ^ (1 > 0)`			`(1 < 0) ^ (0 > 1) ^ (1 > 0)`	

⚠️ Priorités des opérateurs logiques

L'ordre de priorité des opérateurs est, du plus **prioritaire** au moins prioritaire : NOT > AND > XOR > OR

Et l'usage de **parenthèses** sert à modifier cet ordre et forcer une autre évaluation.

➤ Exercice 6

Observez les exemples de code Python ci-dessous, puis déterminez quel(s) message(s) sera (seront) affiché(s). Si rien n'est affiché selon vous, écrire simplement le mot « rien ».

N°	Code Python	Message(s) affiché(s) (à compléter)

1	`age = 10`
	`taille = 1.55`
	`if age > 7 and age < 14 and taille > 1.25 :`
	`    print('Kart 160cc')`	

2	`age = 10`
	`taille = 1.25`
	`if age > 7 and age < 14 and taille > 1.25 :`
	`    print('Kart 160cc')`	

3	`train = True`
	`voiture = False`
	`if train or voiture:`
	`    print('Arrivé(e) à destination')`	

4	`vrai = True`
	`if vrai == True:`
	`    vrai = False`
	`if not vrai == True:`
	`    print('post-verité')`	

5	`cb = True`
	`virement = True`
	`if cb ^ virement:`
	`    print('Paiement effectué')`
	`else:`
	`    print('Paiement annulé')`	

6	`moy = 14`
	`parascol = False`
	`benevol = 50`
	`if moy > 14 and parascol == True or benevol > 40:`
	`    print('Bourse accordée')`
	`else:`
	`    print('Bourse refusée')`	

7	`auth = False`
	`co = True`
	`sms = True`
	`app = True`
	`if (auth == True and co == True) or (sms ^ app):`
	`    print('Streaming OK')`
	`else:`
	`    print('Forbidden')`	

8	`print( True or False and False )`	
9	`print( (True or False) and False )`	