

Logique, ensembles et listes Python

Le programme officiel de maths demande trois choses **transversales**, qui servent dans tous les chapitres : le **vocabulaire des ensembles**, la **logique** (implication, contraposée, absurde) et les **listes en Python**. Cette page les rassemble avec des exemples courts.

Tu y trouveras des astuces , des pièges et un peu d'humour .

Comment utiliser cette page

- Lis chaque partie une fois, puis fais les exercices en cachant les corrections.
- Retiens surtout les **phrases-types** : elles reviennent dans les corrections de tous les chapitres.
- Reviens ici dès qu'un mot te bloque dans un énoncé (« réciproque », « contre-exemple », « par l'absurde »...).

1. Le vocabulaire des ensembles

- $x \in A$: x **appartient** à l'ensemble A . $x \notin A$: il n'y appartient pas.
- $A \subset B$: A est **inclus** dans B (tout élément de A est dans B).
- $A \cap B$: l'**intersection** (les éléments communs). $A \cup B$: la **réunion** (les éléments d'au moins un des deux).
- $\bar{A}$: le **complémentaire** de A (ce qui n'est pas dans A , dans l'ensemble de référence).
- $\emptyset$: l'**ensemble vide**. $A \setminus B$: les éléments de A qui ne sont pas dans B .
- $A \times B$: le **produit cartésien**, l'ensemble des couples $(a; b)$ avec $a \in A$ et $b \in B$.

Exemple (intervalles). $A = [1; 5]$ et $B =]3; 8]$. Alors $A \cap B =]3; 5]$ et $A \cup B = [1; 8]$.

Exemple (complémentaire). Dans $\mathbb{R}$, le complémentaire de $] - \infty; 2]$ est $]2; +\infty[$.

Exemple (produit cartésien). $\{1; 2\} \times \{a; b\} = \{(1; a); (1; b); (2; a); (2; b)\}$: $2 \times 2 = 4$ couples.

Pièges :

- Un **élément** appartient ($\in$) ; un **ensemble** est inclus ($\subset$). On écrit $2 \in \{1; 2\}$ et $\{2\} \subset \{1; 2\}$.
- Le **couple** $(1; 2)$ est **ordonné** : $(1; 2) \neq (2; 1)$. L'ensemble $\{1; 2\}$ ne l'est pas.
- Un crochet **ouvert** ne contient pas la borne : $]3; 8]$ ne contient pas 3 .

Confondre $\in$ et $\subset$, c'est dire « le chat est inclus dans la maison » : on comprend, mais le prof grince.

2. Implication, réciproque, contraposée, équivalence

- **Implication** $P \Rightarrow Q$: « si P , alors Q ». P est une condition **suffisante** pour Q ; Q est une condition **nécessaire** pour P .
- **Réciproque** de $P \Rightarrow Q$: $Q \Rightarrow P$. Elle n'a **aucune raison** d'être vraie en même temps.
- **Contraposée** de $P \Rightarrow Q$: $\text{non } Q \Rightarrow \text{non } P$. Elle est **toujours équivalente** à l'implication de départ.
- **Équivalence** $P \iff Q$: les deux implications sont vraies ($P \Rightarrow Q$ et $Q \Rightarrow P$).

Exemple. « Si $x = 2$, alors $x^2 = 4$ » est vraie. Sa **réciproque** « si $x^2 = 4$, alors $x = 2$ » est **fausse** : $x = -2$ convient aussi.

Exemple. « Si n est un multiple de 4, alors n est pair » est vraie. Sa **contraposée** « si n est impair, alors n n'est pas un multiple de 4 » est vraie aussi.

Négation et quantificateurs. $\forall$ se lit « pour tout » et $\exists$ se lit « il existe ». Pour nier une phrase, on **change le quantificateur** et on **nie la propriété** :

- la négation de « $\forall x \in \mathbb{R}, x^2 > x$ » est « $\exists x \in \mathbb{R}, x^2 \leq x$ » ;
- la négation de « $\exists n \in \mathbb{N}, n^2 = 2$ » est « $\forall n \in \mathbb{N}, n^2 \neq 2$ ».

Contre-exemple. Pour **réfuter** une phrase commençant par « pour tout », **un seul** exemple suffit. « Tous les nombres premiers sont impairs » est fausse : 2 est premier et pair.

Exemple. « $\forall x \in \mathbb{R}, x^2 > x$ » est fausse : pour $x = \frac{1}{2}$, $x^2 = \frac{1}{4} < \frac{1}{2}$.

Astuce. Devant une affirmation « pour tout », **teste d'abord 0, 1 et -1** . Un contre-exemple se cache souvent là.

Pièges :

- **Vérifier sur des exemples ne prouve rien** : cela ne fait que rendre la phrase plausible.
- La négation de « tous » n'est **pas** « aucun » : c'est « **au moins un** ne vérifie pas ».
- Ne confonds pas **réciproque** et **contraposée**. La contraposée a toujours la même valeur de vérité que l'implication.

« Tous les cygnes que j'ai vus sont blancs » ne prouve pas que tous les cygnes sont blancs. Il suffit d'un cygne noir.

3. Trois raisonnements à connaître

La disjonction de cas. On sépare toutes les situations possibles et on traite chacune.

Exemple. Montrer que, pour tout entier n , $n(n+1)$ est pair.

Si n est pair, $n(n+1)$ contient le facteur pair n : le produit est pair.

Si n est impair, $n+1$ est pair : le produit est pair.

Dans les deux cas, $n(n+1)$ est pair. $\square$

Le raisonnement par contraposée. Pour prouver $P \Rightarrow Q$, on prouve $\text{non } Q \Rightarrow \text{non } P$.

Exemple. Montrer que si n^2 est pair, alors n est pair.

Contraposée : si n est impair, alors n^2 est impair. Avec $n = 2k + 1$: $n^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$, qui est impair. $\square$

Le raisonnement par l'absurde. Pour prouver P , on suppose $\text{non } P$ et on aboutit à une contradiction.

Exemple. Montrer que $\sqrt{2}$ est irrationnel.

Supposons $\sqrt{2} = \frac{p}{q}$ avec p et q entiers, la fraction étant **irréductible**.

Alors $p^2 = 2q^2$, donc p^2 est pair, donc p est pair (contraposée ci-dessus) : $p = 2m$.

Alors $4m^2 = 2q^2$, soit $q^2 = 2m^2$, donc q est pair.

p et q sont tous les deux pairs : la fraction n'est pas irréductible. C'est absurde. Donc $\sqrt{2}$ est irrationnel. $\square$

Astuce. Dans un énoncé, les mots « **montrer qu'il n'existe pas** » ou « **montrer que... est impossible** » appellent presque toujours un raisonnement par l'absurde.

En Terminale : récurrence et quantificateurs. Une démonstration par récurrence prouve une phrase du type « $\forall n \in \mathbb{N}, \mathcal{P}(n)$ » en deux étapes : l'**initialisation** ($\mathcal{P}(0)$ est vraie) et l'**hérédité** (pour tout n , $\mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$).

Attention à la négation. La négation de « la suite est croissante » n'est pas « elle est décroissante » : c'est « il existe un rang n tel que $u_{n+1} < u_n$ ». Une suite peut n'être ni l'un ni l'autre.

4. Les listes en Python

Une **liste** range plusieurs valeurs entre crochets. Les éléments sont numérotés à partir de **0**.

```
L = [3, 1, 4, 1, 5]
print(L[0])      # affiche 3 (premier élément)
print(L[-1])     # affiche 5 (dernier élément)
print(len(L))    # affiche 5 (nombre d'éléments)
L[1] = 7         # modifie le deuxième élément
print(L)         # affiche [3, 7, 4, 1, 5]
```

Ajouter un élément, supprimer, extraire.

```
L = [3, 1, 4]
L.append(9)       # ajoute à la fin : [3, 1, 4, 9]
L.remove(1)       # supprime la première valeur 1 : [3, 4, 9]
print(L[0:2])     # affiche [3, 4] (indices 0 et 1, pas le 2)
print(sum(L), max(L), min(L)) # affiche 16 9 3
```

Parcourir une liste. On peut prendre les éléments un par un, ou passer par les indices.

```
L = [3, 1, 4]
for x in L:
    print(x)       # affiche 3, puis 1, puis 4

for i in range(len(L)):
    print(i, L[i]) # affiche 0 3, puis 1 1, puis 2 4
```

Créer une liste par compréhension. C'est la façon la plus courte de fabriquer une liste.

```
carres = [k ** 2 for k in range(1, 6)]
print(carres) # affiche [1, 4, 9, 16, 25]

pairs = [k for k in range(10) if k % 2 == 0]
print(pairs)  # affiche [0, 2, 4, 6, 8]
```

Un exemple de suite. On range les termes de $u_0 = 2$, $u_{n+1} = 3u_n - 1$ dans une liste :

```
u = [2]
for n in range(5):
    u.append(3 * u[-1] - 1)
print(u) # affiche [2, 5, 14, 41, 122, 365]
```

Pièges :

- **Le premier indice est 0** : une liste de 5 éléments va de `L[0]` à `L[4]`.
- `L[0:2]` s'arrête **avant** l'indice 2.
- `range(1, 6)` donne 1, 2, 3, 4, 5 : le dernier nombre **n'est pas** atteint.
- Une liste **se modifie en place** : `L.append(9)` ne renvoie rien, il faut écrire `L.append(9)` et non `L = L.append(9)`.

Compter à partir de 0 est la seule raison pour laquelle un programmeur ne rate pas son bus : il arrive toujours en retard d'un cran.

5. Quatre exercices**Exercice 1 (★) : intervalles et ensembles**

Soient $A = [-2; 3]$ et $B =]1; 6[$.

1. Déterminer $A \cap B$ et $A \cup B$.
2. Déterminer le complémentaire de A dans $\mathbb{R}$.

Exercice 2 (★★) : réciproque, contraposée, négation

On considère l'implication « si $x > 2$, alors $x^2 > 4$ ».

1. Énoncer sa réciproque. Est-elle vraie ?
2. Énoncer sa contraposée. Est-elle vraie ?
3. Écrire la négation de « $\forall x \in \mathbb{R}, x^2 > x$ » puis dire si la phrase de départ est vraie.

Exercice 3 (★★) : disjonction de cas et absurde

1. Montrer que, pour tout entier n , $n^2 + n$ est pair.
2. Montrer par l'absurde qu'il n'existe pas d'entier n tel que $n^2 + 1$ soit divisible par 4.

Exercice 4 (★★) : listes et suites

On définit $u_0 = 1$ et $u_{n+1} = 2u_n + 3$.

1. Écrire un script Python qui range $u_0, \dots, u_5$ dans une liste.
2. Quelle liste obtient-on ? En déduire u_5 .

```
u = [1]
for n in range(5):
    u.append(2 * u[-1] + 3)
print(u)
```

À retenir

- **Ensembles** : $\in$ pour un élément, $\subset$ pour un ensemble ; $\cap$ pour « et », $\cup$ pour « ou ».
- **Implication** : la contraposée a toujours la même valeur de vérité ; la **réci-proque** non.
- **Pour réfuter** un « pour tout » : **un seul contre-exemple**.
- **Pour nier** : $\forall \leftrightarrow \exists$ et on nie la propriété.
- **Python** : indices à partir de 0, `append`, parcours avec `for`, compréhension.