

Logique, ensembles et listes Python

Le programme officiel de maths comporte trois parties **transversales**, qui servent dans tous les chapitres : « Vocabulaire ensembliste et logique », « Algorithmique et programmation » (avec la notion de **liste**) et « Automatismes » (traités dans la page qui leur est consacrée). Cette page rassemble les deux premières, 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).
- $\bar{A}$ ou $E \setminus A$: le **complémentaire** de A dans l'ensemble de référence E (ce qui est dans E mais pas dans A).
- $\emptyset$: l'**ensemble vide**. $A \setminus B$: les éléments de A qui ne sont pas dans B .
- $\text{Card}(A)$: le **cardinal** de A , c'est-à-dire son nombre d'éléments (pour un ensemble fini).
- $A \times B$: le **produit cartésien**, l'ensemble des couples $(a; b)$ avec $a \in A$ et $b \in B$.

Les ensembles de nombres.

Notation	Ensemble	Exemples
$\mathbb{N}$	entiers naturels	$0; 1; 2; \dots$
$\mathbb{Z}$	entiers relatifs	$-3; 0; 5$
$\mathbb{D}$	nombres décimaux	$0,25; -1,5$
$\mathbb{Q}$	nombres rationnels (quotients d'entiers)	$\frac{1}{3}; -\frac{7}{4}$
$\mathbb{R}$	nombres réels	$\sqrt{2}; \pi$

Ces ensembles sont emboîtés : $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{D} \subset \mathbb{Q} \subset \mathbb{R}$.

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 (cardinal). $A = \{2 ; 4 ; 6\}$ a pour cardinal $\text{Card}(A) = 3$, et $\text{Card}(\emptyset) = 0$.

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

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. Connecteurs, implication, réciproque, contraposée, équivalence

- « **et** » : la proposition « P et Q » est vraie quand P et Q sont vraies toutes les deux. Elle correspond à l'intersection $\cap$.
- « **ou** » : la proposition « P ou Q » est vraie quand l'une **au moins** est vraie (le « ou » est **inclusif** : les deux peuvent être vraies). Elle correspond à la réunion $\cup$.

Exemples. « $x \geq -2$ et $x \leq 3$ » signifie $x \in [-2 ; 3]$. « $x < -2$ ou $x > 3$ » signifie $x \in]-\infty ; -2[\cup]3 ; +\infty[$. Pour $x = 4$, la proposition « $x > 3$ ou x est pair » est vraie, et elle est vraie aussi pour $x = 6$ (les deux sont vraies).

- **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 $Q \Rightarrow P$). On dit aussi « P si et seulement si Q ».

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.

Utiliser l'équivalence dans un raisonnement. Quand on résout une équation, chaque ligne doit être **équivalente** à la précédente : on ne perd ni n'ajoute de solution.

$$3x - 5 = 7 \iff 3x = 12 \iff x = 4$$

Attention : élever au carré donne seulement une **implication**. De $x = -1$, on tire $x^2 = 1$, mais $x^2 = 1$ n'entraîne pas $x = -1$: $x = 1$ convient aussi.

Le statut des égalités et des lettres.

- Une **identité** est une égalité vraie pour **toutes** les valeurs de la lettre : $(x + 1)^2 = x^2 + 2x + 1$ est vraie pour tout réel x .
- Une **équation** est une égalité que l'on cherche à rendre vraie : $x^2 = 4$ n'est vraie que pour $x = -2$ et $x = 2$.
- Une **variable** prend des valeurs quelconques d'un ensemble (dans $f(x) = 3x + 1$, x est la variable). Une **inconnue** est la valeur que l'on cherche. Un **paramètre** est une lettre fixée dans un problème, mais qui peut changer d'un problème à l'autre : dans $mx + 2 = 0$, x est l'inconnue et m un paramètre.

Quantificateurs, en français. On dit « **pour tout** » et « **il existe** ». Les symboles $\forall$ et $\exists$ existent, mais ils **ne sont pas exigibles** : tu peux (et tu dois) écrire en français.

Négation d'une proposition quantifiée. Pour nier une phrase, on **change le quantificateur** et on **nie la propriété** :

- la négation de « pour tout réel x , $x^2 > x$ » est « **il existe** un réel x tel que $x^2 \leq x$ » ;
- la négation de « il existe un entier n tel que $n^2 = 2$ » est « **pour tout** entier n , $n^2 \neq 2$ ».

Quantifications implicites. Dans une phrase conditionnelle, le « pour tout » est souvent **caché** : « si $x > 2$, alors $x^2 > 4$ » signifie « **pour tout réel x , si $x > 2$ alors $x^2 > 4$** ». Pour la nier, il faut le rétablir : « il existe un réel $x > 2$ tel que $x^2 \leq 4$ ».

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. « Pour tout réel x , $x^2 > x$ » 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 **réciroque** 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. Raisonnements : disjonction des cas, contraposée, absurde

La disjonction des 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.

4. Notion de liste (algorithmique et programmation)

En Première, la seule notion nouvelle est celle de **liste**. Le reste consolide ce que tu connais : variables, instructions conditionnelles, boucles, fonctions.

- **Affectation.** En langage naturel, on écrit $x \leftarrow 3$ (« x prend la valeur **3** ») et $n \leftarrow n + 1$ (« n prend la valeur $n + 1$ »). En Python : `x = 3` et `n = n + 1`.
- **Types de variables.** Un entier (`int`), un nombre à virgule (`float`), un texte (`str`), un booléen (`bool`, `True` ou `False`), une liste (`list`). `3` et `3.0` n'ont pas le même type, et `type(x)` donne le type de `x`.
- **Programmation modulaire.** On découpe une tâche en **fonctions** simples, que l'on réutilise : une fonction `def` reçoit des arguments et renvoie un résultat avec `return`.

```
def est_pair(n):
    return n % 2 == 0

print(est_pair(6), est_pair(7))    # affiche True False
```

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]
```

Générer une liste : en extension, par ajouts successifs, en compréhension.

```
A = [2, 4, 6]          # en extension

B = []                # par ajouts successifs
for k in range(1, 4):
    B.append(2 * k)    # B vaut [2, 4, 6]

C = [2 * k for k in range(1, 4)]  # en compréhension : [2, 4, 6]
```

Ajouter, 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]
L.pop(0)             # supprime l'élément d'indice 0 : [4, 9]
L.insert(1, 8)        # insère 8 à l'indice 1 : [4, 8, 9]
print(L[0:2])         # affiche [4, 8] (indices 0 et 1, pas le 2)
print(sum(L), max(L), min(L))  # affiche 21 9 4
```

Parcourir une liste. On peut prendre les éléments un par un (itérer sur les éléments), 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
```

Liste en compréhension et ensemble. Une liste en compréhension se lit comme un ensemble défini par une **condition**. La liste `[k for k in range(1, 11) if k % 3 == 0]` correspond à l'ensemble $\{k \in \mathbb{N} \mid 1 \leq k \leq 10 \text{ et } k \text{ multiple de } 3\} = \{3; 6; 9\}$.

```
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]

M = [k for k in range(1, 31) if k % 3 == 0 and k % 5 == 0]
print(M, len(M))   # affiche [15, 30] 2 (condition avec et)
```

La condition d'une liste en compréhension utilise les connecteurs de la partie 2 : **and** pour « et », **or** pour « ou » (inclusif), **not** pour la négation.

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.

Histoire des mathématiques. De nombreux textes témoignent d'une préoccupation algorithmique au long de l'Histoire. Quand un texte historique décrit une méthode de calcul, on peut la **transformer en algorithme**, voire en programme, ou faire l'inverse : ce changement de registre donne du sens au formalisme. Le programme cite par exemple le calcul de la racine carrée chez Héron d'Alexandrie (voir l'exercice 8).

5. Huit 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, **en français**, la négation de « pour tout réel x , $x^2 > x$ » 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 .

Exercice 5 (★) : connecteurs, identités, quantifications implicites

1. Traduire avec « et » ou « ou » : $x \in [-2; 3]$; puis $x \in]-\infty; -2[\cup]3; +\infty[$.
2. Les égalités suivantes sont-elles des identités ou des équations ? (a) $(x + 1)^2 = x^2 + 2x + 1$; (b) $x^2 = 4$; (c) $2(x + 3) = 2x + 6$.
3. Dans $mx + 2 = 0$, quel est le rôle de x ? de m ? Résoudre cette équation pour $m \neq 0$.
4. Rétablir la quantification cachée dans « si $x > 2$, alors $x^2 > 4$ », puis écrire la négation de cette phrase. Cette négation est-elle vraie ?

Exercice 6 (★★) : résoudre par équivalences

1. Résoudre $2x + 3 = 7$ en écrivant des équivalences.
2. Résoudre $(x - 1)(x + 2) = 0$ en justifiant chaque équivalence.
3. On considère l'équation $\sqrt{x + 2} = x$. Montrer que $\sqrt{x + 2} = x \Rightarrow x^2 = x + 2$, résoudre $x^2 = x + 2$, puis conclure : l'implication précédente est-elle une équivalence ?

Exercice 7 (★★) : listes, ensembles et cardinaux

On note A l'ensemble des multiples de 3 compris entre 1 et 30, et B celui des multiples de 5 compris entre 1 et 30.

1. Écrire en Python la liste A en compréhension. Quel est $\text{Card}(A)$?
2. Déterminer $A \cap B$ et $A \cup B$ à la main, puis donner leurs cardinaux.
3. Écrire en compréhension la liste de $A \cap B$, puis celle de $A \cup B$ (avec « ou »).
4. Pour la liste U de $A \cup B$, quelles instructions suppriment 15 puis ajoutent 33 à la fin ? Que vaut $\text{len}(U)$ ensuite ?

Exercice 8 (★★) : un algorithme de calcul de racine carrée

Pour approcher $\sqrt{2}$, on utilise la méthode attribuée à Héron d'Alexandrie : $u_0 = 1$ et, pour tout n ,

$$u_{n+1} = \frac{u_n + \frac{2}{u_n}}{2}.$$

1. Écrire cet algorithme en langage naturel, avec le symbole $\leftarrow$, pour calculer u_3 .
2. Écrire un script Python qui range $u_0, \dots, u_3$ dans une liste.
3. Calculer u_1, u_2, u_3 sous forme de fractions. Vérifier que u_3 approche $\sqrt{2}$ à moins de 10^{-5} .

À retenir

- **Ensembles** : $\in$ pour un élément, $\subset$ pour un ensemble ; $\cap$ pour « et », $\cup$ pour « ou » (inclusif) ; **Card** pour le nombre d'éléments.
- **Implication** : la contraposée a toujours la même valeur de vérité ; la **réciproque** non. Une résolution se fait par **équivalences**.
- **Pour réfuter** un « pour tout » : **un seul contre-exemple**.
- **Pour nier** : « pour tout » devient « il existe » (et inversement), et on nie la propriété. Pense aux « pour tout » **cachés**.
- **Python** : indices à partir de 0, **append**, **remove**, parcours avec **for**, liste en compréhension (condition = logique), fonctions avec **def**.