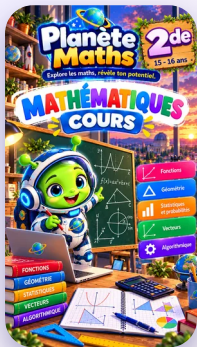


Logique et algorithmique : cours de maths 2de à télécharger en PDF



Le livre de cours de maths 2de

Tout le programme en un seul PDF de 99 pages, à imprimer ou à garder sur ta tablette.

[Voir le livre](#)

« Si tu as ton badge, alors tu peux entrer. » Cette phrase paraît anodine, pourtant elle cache tout un art : savoir ce qu'elle affirme, ce qu'elle n'affirme pas, et comment prouver ou réfuter une affirmation. C'est le travail de la **logique**. Dans ce chapitre, tu vas aussi donner des ordres précis à un ordinateur : c'est l'**algorithmique**. Les deux sont liés, car un programme qui fonctionne n'est rien d'autre qu'un raisonnement soigneusement écrit.

1. Propositions, « et », « ou », négation

D

Proposition

Une proposition est un énoncé mathématique qui est soit **vrai**, soit **faux**, jamais les deux. Exemples : « **17** est un nombre premier » (vraie), « **$2 + 2 = 5$** » (fausse). « Quel beau triangle ! » n'est pas une proposition.

On combine les propositions avec deux petits mots. « P et Q » est vraie quand les deux sont vraies. « P ou Q » est vraie quand **au moins une** des deux est vraie : en mathématiques, le « ou » est **inclusif**, il n'exclut pas le cas où les deux sont vraies (contrairement au « fromage ou dessert » du restaurant).

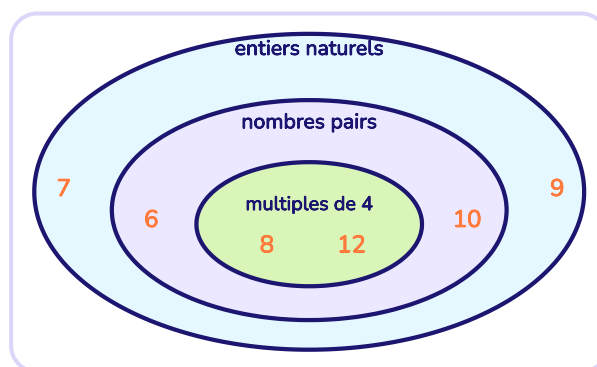
La **négation** de P , notée « non P », est vraie exactement quand P est fausse. Nier « $x > 3$ » donne « $x \leq 3$ » et non « $x < 3$ » : le cas $x = 3$ ne doit pas disparaître.

2. Implication, réciproque et contraposée

D Implication

L'implication « $P \Rightarrow Q$ », qui se lit « si P , alors Q », affirme que dès que P (l'**hypothèse**) est vraie, Q (la **conclusion**) l'est aussi. Sa **réciproque** est « $Q \Rightarrow P$ ». Sa **contraposée** est « non $Q \Rightarrow$ non P ».

Une implication est fausse dans un seul cas : quand l'hypothèse est vraie et la conclusion fausse. Un point essentiel : **une implication et sa contraposée sont toujours vraies ou fausses ensemble**, alors que la réciproque est une affirmation différente, qu'il faut examiner à part.



Sur la figure, tout multiple de 4 est pair : l'implication « si n est multiple de 4, alors n est pair » est vraie. Mais 6 est pair sans être multiple de 4 : la réciproque est fausse.

E**Exemple : réciproque et contraposée**

Implication : « si un entier est multiple de **9**, alors il est multiple de **3** ». Elle est vraie, car $9k = 3 \times 3k$.

Réciproque : « si un entier est multiple de **3**, alors il est multiple de **9** ». Elle est fausse : **12** est multiple de **3** mais pas de **9**.

Contraposée : « si un entier n'est pas multiple de **3**, alors il n'est pas multiple de **9** ». Elle est vraie, comme l'implication de départ.

**Erreur classique**

Confondre une implication et sa réciproque. « S'il pleut, le sol est mouillé » est vraie ; « si le sol est mouillé, il pleut » ne l'est pas (un arrosage suffit).

3. Équivalence**D****Équivalence**

On écrit $P \Leftrightarrow Q$ (« P si et seulement si Q ») quand $P \Rightarrow Q$ et $Q \Rightarrow P$ sont toutes les deux vraies. Les deux propositions sont alors vraies ensemble ou fausses ensemble.

M**Méthode : démontrer une équivalence**

1. Démontre l'implication directe $P \Rightarrow Q$.
2. Démontre la réciproque $Q \Rightarrow P$.
3. Conclut : $P \Leftrightarrow Q$.

E**Exemple : résoudre par équivalences**

Pour un réel x : $3x - 5 = 10 \Leftrightarrow 3x = 15 \Leftrightarrow x = 5$. Chaque étape peut se lire dans les deux sens, donc **5** est l'unique solution.

En revanche, $x^2 = 16 \Leftrightarrow x = 4$ est **fausse** : de $x = 4$ on déduit $x^2 = 16$, mais $x^2 = 16$ est aussi vérifiée par $x = -4$.

4. Les quantificateurs « pour tout » et « il existe »

Pour parler de tous les éléments d'un ensemble, ou de l'un d'entre eux, on dispose de deux symboles.

D**Quantificateurs**

$\forall$ signifie « **pour tout** » : « $\forall x \in \mathbb{R}, x^2 \geq 0$ » dit que le carré de n'importe quel réel est positif ou nul. $\exists$ signifie « **il existe (au moins un)** » : « $\exists n \in \mathbb{N}, n + 3 = 10$ » est vraie, avec $n = 7$.

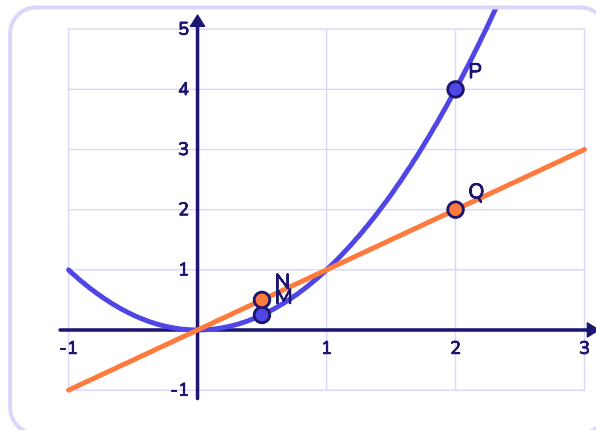
P**Négation avec quantificateurs**

La négation de « $\forall x, P(x)$ » est « $\exists x, \text{non } P(x)$ ». La négation de « $\exists x, P(x)$ » est « $\forall x, \text{non } P(x)$ ».

Exemple dans la vie courante : nier « tous les élèves du groupe ont un téléphone » donne « au moins un élève du groupe n'a pas de téléphone », et non « aucun élève n'a de téléphone ».

E**Exemple : une proposition en $\forall$ à tester**

Considérons « $\forall x \in \mathbb{R}, x^2 \geq x$ ». Pour $x = 3 : 9 \geq 3$, vrai ; pour $x = -2 : 4 \geq -2$, vrai. Mais pour $x = 0,5 : x^2 = 0,25$ et $0,25 < 0,5$. La proposition est donc **fausse**. Sa négation, « $\exists x \in \mathbb{R}, x^2 < x$ », est vraie.



5. Contre-exemple et modes de raisonnement

P

Contre-exemple

Pour **réfuter** une proposition du type « $\forall x, P(x)$ », il suffit d'exhiber **un seul x** pour lequel $P(x)$ est fausse : c'est un **contre-exemple**. À l'inverse, des centaines d'exemples qui marchent ne **prouvent** jamais une proposition en « pour tout ».

Pour **prouver** « pour tout entier n », on prend un entier n **quelconque** et on raisonne avec des lettres, pas avec des valeurs particulières.

E

Exemple : disjonction de cas

Montrons que, pour tout entier n , $n^3 - n$ est pair.

Si n est pair, n^3 est pair, et la différence de deux pairs $n^3 - n$ est paire. Si n est impair, n^3 est impair, et la différence de deux impairs $n^3 - n$ est paire. Dans les deux cas $n^3 - n$ est pair, et comme tout entier est pair ou impair, la propriété est démontrée.

E

Exemple : raisonnement par l'absurde

Montrons qu'il n'existe pas de plus petit réel strictement positif. Supposons qu'il en existe un, noté a . Alors $\frac{a}{2}$ est un réel strictement positif et $\frac{a}{2} < a$, ce qui contredit le choix de a . L'hypothèse était donc absurde : il n'existe pas de plus petit réel strictement positif.



Astuce de Zyro

Sur ma planète, on dit : « un seul grain de poussière dans l'engrenage suffit à l'arrêter. » C'est exactement le rôle du contre-exemple : une seule valeur qui cloche réfute un « pour tout ».

6. Variables et affectations

Une **variable** est une case de mémoire portant un nom, qui contient une valeur. L'**affectation** range une valeur dans une variable ; on la note $a \leftarrow 7$ en pseudo-code et $a = 7$ en Python. Attention : « = » n'est pas ici une égalité mathématique, mais un **ordre** : « calcule le membre de droite, puis range le résultat dans la variable de gauche ».

C'est pourquoi l'instruction $n = n + 1$, absurde comme équation, est parfaitement légale : elle augmente n de 1. Pour suivre un programme à la main, on dresse un **tableau de suivi** qui donne les valeurs des variables après chaque ligne.

E**Exemple : suivre des affectations**

$a = 7$

$b = a * 2$

$a = b - a$

$b = a + 1$

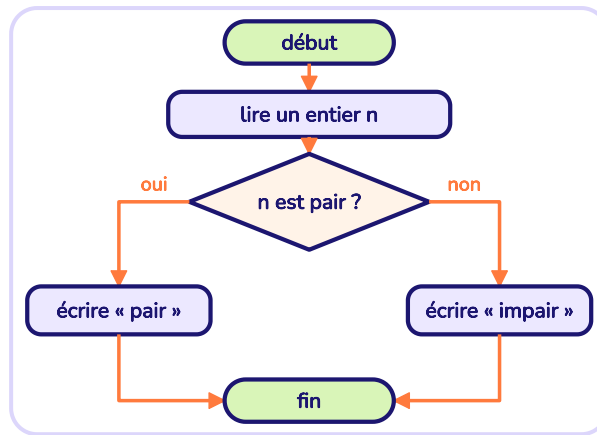
Après la ligne	a	b
a = 7	7	—
b = a * 2	7	14
a = b - a	7	14
b = a + 1	7	8

À la fin, $a = 7$ et $b = 8$. Remarque : à la troisième ligne, a reçoit $14 - 7 = 7$, donc garde la même valeur.

Pour **échanger** deux variables a et b , il faut une variable auxiliaire : $t = a$, puis $a = b$, puis $b = t$. Sans elle, on écraserait une valeur.

7. Conditions et boucles

Une **condition** « si... sinon... » permet à un programme de choisir. En Python on écrit **if** suivi d'un test, deux-points, puis un bloc **décalé vers la droite** (l'indentation compte !). Le test utilise **==** (égal), **!=** (différent), **<**, **>**, et se combine avec **and**, **or**, **not**, qui sont les « et », « ou » et la négation de la partie 1.



```

n = int(input())
if n % 2 == 0:
    print("pair")
else:
    print("impair")

```

L'opérateur **%** donne le reste de la division euclidienne : **17 % 5** vaut **2**, et un entier est pair exactement quand **n % 2** vaut **0**. L'opérateur **//** donne le quotient entier.

Une **boucle pour** répète un bloc un nombre connu de fois : **for i in range(1, 6)**: fait prendre à **i** les valeurs **1, 2, 3, 4, 5** (la borne **6** est exclue). Une **boucle tant que** répète tant qu'une condition reste vraie : **while c < 300**:

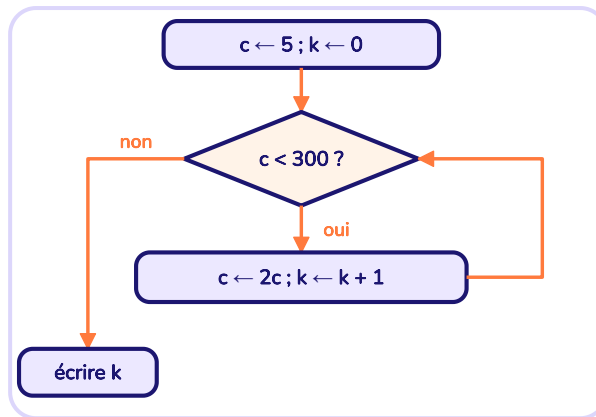
E Exemple : boucle pour

```

S = 0
for i in range(1, 6):
    S = S + 2 * i - 1
print(S)

```

Les valeurs successives de **2i - 1** sont **1, 3, 5, 7, 9** et **S** devient **1, 4, 9, 16, 25**. Le programme affiche **25**, c'est-à-dire **5²**.



E Exemple : boucle tant que

L'organigramme ci-dessus se traduit ainsi :

```
c = 5
k = 0
while c < 300:
    c = 2 * c
    k = k + 1
print(k)
```

La variable c prend les valeurs **5, 10, 20, 40, 80, 160, 320**. Dès que $c = 320$, le test $c < 300$ devient faux et la boucle s'arrête, après **6** passages. Le programme affiche **6**.



Erreur classique

Écrire une boucle « tant que » dont la condition ne devient jamais fausse : le programme tourne sans fin. Vérifie toujours qu'une variable du test évolue dans la boucle.

8. Programmer en Python : fonctions et tests de conjectures

Une **fonction** regroupe des instructions sous un nom et renvoie un résultat avec **return**. Elle se définit avec **def**.

```
def est_multiple(n, d):  
    return n % d == 0
```

L'appel **est_multiple(21, 7)** renvoie **True** (vrai) et **est_multiple(22, 7)** renvoie **False** (faux). Le résultat est un **booléen** : c'est la valeur de vérité d'une proposition.

La programmation sert aussi à **chercher un contre-exemple**. Testons la conjecture « pour tout entier $n \geq 1$, $n^2 \leq 2^n$ » :

```
n = 1  
while n ** 2 <= 2 ** n:  
    n = n + 1  
print(n)
```

La boucle s'arrête au premier n qui **ne vérifie pas** l'inégalité. Pour $n = 1$ et $n = 2$ elle est vraie ($1 \leq 2$ et $4 \leq 4$), mais pour $n = 3$ on a $9 > 8$: le programme affiche **3**, et $n = 3$ est un contre-exemple. La conjecture est fausse. Un programme peut ainsi **réfuter** une proposition « pour tout », mais jamais la prouver, car il ne teste qu'un nombre fini de valeurs.

À retenir

- Une **implication** $P \Rightarrow Q$ a pour **réciroque** $Q \Rightarrow P$ (autre affirmation) et pour **contraposée** « non $Q \Rightarrow$ non P » (équivalente à l'implication).
- $P \Leftrightarrow Q$ signifie que les deux implications sont vraies : P et Q sont vraies ensemble ou fausses ensemble.
- $\forall$ = « pour tout », $\exists$ = « il existe ». Nier $\forall$ donne $\exists$ non, nier $\exists$ donne $\forall$ non.
- Un seul **contre-exemple** réfute un « pour tout » ; des exemples ne le prouvent jamais.
- En Python, `=` est une affectation, `==` un test d'égalité ; `%` donne le reste et `//` le quotient.
- **for** répète un nombre connu de fois (`range(1, 6)` exclut 6) ; **while** répète tant que la condition est vraie, et doit pouvoir s'arrêter.

Le livre de cours de maths 2de

Tout le programme en un seul PDF de 99 pages, à imprimer ou à garder sur ta tablette.

Voir le livre

Zyro te conseille la suite

 Exercices de maths

Logique et algorithmique : exercices de maths 2de

 Contrôles de maths

Logique et algorithmique : contrôle de maths 2de

 QCM de maths

Logique et algorithmique : QCM de maths 2de

 Exercices de maths

Nombres et calculs : exercices de maths 2de

 Exercices de maths

Nombres et ensembles de nombres : exercices de maths 2de

 Contrôles de maths

Nombres et calculs : contrôle de maths 2de

Zyro vous emmène sur nos sites partenaires

Embarquez pour d'autres planètes du savoir et leurs applis.

Décollage