# Plaquette 3 – Boucles non bornées, seuils et terminaison

## Algorithmique et Python — Première spé maths

Corrections détaillées

### Boîte à outils

**Boucle non bornée.**

```
while condition:
    instruction
```

La boucle tourne **tant que** la condition est vraie. Si la condition ne devient jamais fausse, le programme **ne s'arrête pas**.

**Schéma d'une boucle de seuil.**

```
u = u0
```

```
n = 0
while u < seuil:
    u = suivant(u)
    n = n + 1
print(n)
```

**Justifier la terminaison.** Il faut établir que la condition finira par être fausse :

| Situation | Terminaison |
|---|---|
| suite croissante de limite $+\infty$ | garantie pour tout seuil |
| suite décroissante de limite 0 | garantie pour tout seuil $> 0$ |
| suite **majorée** par le seuil | **jamais** — boucle infinie |
| suite oscillante | à examiner au cas par cas |

**Piège central.** Une suite croissante **majorée** converge vers une limite $\ell$ ; un seuil supérieur à $\ell$ ne sera **jamais** atteint. Il faut donc toujours comparer le seuil à la **limite**.

**Calcul direct d'un seuil géométrique.** Pour $u_n = u_0 q^n$ :

$$u_0 q^n > s \Leftrightarrow n > \frac{\ln(s/u_0)}{\ln q} \quad (q > 1)$$

Si $0 < q < 1$, le sens de l'inégalité **s'inverse** en divisant par $\ln q < 0$.

**Coût comparé.** Une boucle de seuil coûte $n$ tours ; le logarithme coûte un calcul **constant**. La boucle reste indispensable quand la suite n'a pas de forme explicite.

**Condition d'arrêt : le complémentaire.** Pour obtenir « le premier $n$ tel que $u_n > s$ », la condition de boucle est $u <= s$, pas $u > s$.

## Correction de l'exercice 1 – Boucle de seuil croissante

— **Suite** : $u_n = 2^n$, géométrique croissante.
— **Condition d'arrêt** : $2^n \geq 100$.
— **Puissances de** 2 : $2^6 = 64 < 100$ et $2^7 = 128 \geq 100$ ✓
— **Réponse** : le script affiche 7 ✓

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

— **Par le logarithme** : $n \geq \dfrac{\ln 100}{\ln 2} \approx \dfrac{4{,}6052}{0{,}69315} \approx 6{,}644$, donc $n = 7$ ✓
— **Trace** : $u$ vaut 2, 4, 8, 16, 32, 64, 128 — sept multiplications ✓
— **Terminaison** : $q = 2 > 1$, donc $u_n \to +\infty$ — la boucle s'arrête nécessairement ✓
— **Attention à la condition** : $u <$ *100* fait continuer tant que $u$ est **sous** le seuil ; on sort dès que $u \geq 100$ ✓
— **Variante** : avec *while u <= 100*, le résultat serait identique, car $u$ ne vaut jamais exactement 100 ✓
— **Valeur finale de** $u$ : 128, qu'on pourrait afficher aussi en écrivant *print(n, u)*.

**Réponse.** 7 (avec $u = 128$).

## Correction de l'exercice 2 – Boucle de seuil décroissante

**Script.**

```
u = 1.0
```

```
n = 0
while u >= 0.01:
    u = 0.95 * u
    n = n + 1
print(n)
```

— **Résolution par le logarithme** (avec $\ln 0{,}95 < 0$, sens **inversé**) :

$$n > \frac{\ln 0{,}01}{\ln 0{,}95} \approx \frac{-4{,}6052}{-0{,}051293} \approx 89{,}781$$

— **Réponse** : $n = 90$ ✓
— **Vérifications** : $0{,}95^{89} \approx 0{,}010409 \geq 0{,}01$ ✓ et $0{,}95^{90} \approx 0{,}0098884 < 0{,}01$ ✓
— **Marge étroite** : le seuil est franchi de peu (0,00989 contre 0,01), d'où l'intérêt de **vérifier numériquement** les deux rangs encadrants plutôt que de se fier à l'arrondi de 89,781 ✓
— **Contrôle du rang suivant** : $0{,}95^{91} \approx 0{,}0093939$ — la décroissance est bien de 5 % par rang ✓
— **Terminaison** : $0 < q < 1$ donne $u_n \to 0$, donc la boucle s'arrête pour tout seuil strictement positif ✓
— **Demi-vie** : $\dfrac{\ln 0{,}5}{\ln 0{,}95} \approx 13{,}513$ — la suite est divisée par 2 tous les 13,5 rangs ✓

**Réponse.** $n = 90$ (avec $0{,}95^{90} \approx 0{,}009888 < 0{,}01$).

## Correction de l'exercice 3 – Terminaison en défaut

— **Suite** : $u_0 = 0$ et $u_{n+1} = 0{,}5u_n + 2$.
— **Point fixe** : $0{,}5x + 2 = x$ donne $0{,}5x = 2$, soit $x = 4$.
— **Forme explicite** : avec $v_n = u_n - 4$, on a $v_{n+1} = 0{,}5v_n$ et $v_0 = -4$, donc

$$u_n = 4 - 4 \times 0{,}5^n$$

— **Limite** : $u_n \to 4$, en restant **strictement inférieure** à 4 ✓
— **Conclusion** : la suite est **majorée par** $4 < 5$ — la condition $u < 5$ reste **toujours vraie** et le script **ne se termine jamais** ✓
— **Trace** : $u$ vaut 2 ; 3 ; 3,5 ; 3,75 ; 3,875 ; … — elle s'approche de 4 sans jamais l'atteindre ✓
— **Correction possible** : remplacer le seuil par 3,99, atteignable puisque $3{,}99 < 4$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

— **Calcul du rang** : $4 - 4 \times 0{,}5^n > 3{,}99$ donne $0{,}5^n < 0{,}0025$, soit $n > \dfrac{\ln 0{,}0025}{\ln 0{,}5} \approx 8{,}644$, donc $n = 9$ ✓
— **Vérification** : $u_9 = 4 - 4 \times 0{,}001953 \approx 3{,}99219 > 3{,}99$ ✓ et $u_8 \approx 3{,}98438 < 3{,}99$ ✓
— **Règle absolue** : avant d'écrire une boucle *while*, calculer la **limite** de la suite et vérifier que le seuil est du bon côté ✓
— **Sécurité en pratique** : ajouter un compteur maximal, par exemple *while u < 5 and n < 10000*, évite le blocage ✓

**Réponse.** Non : $u_n = 4 - 4 \times 0{,}5^n$ est majorée par $4 < 5$ — boucle infinie.

## Correction de l'exercice 4 – Seuil sur une suite arithmétique

**Script.**

```
u = 5
```

```
n = 0
while u <= 1000:
    u = u + 17
    n = n + 1
print(n)
```

— **Calcul direct** : $u_n = 5 + 17n > 1000$ donne $17n > 995$, soit

$$n > \frac{995}{17} \approx 58{,}529$$

— **Réponse** : $n = 59$ ✓
— **Vérifications** : $u_{58} = 5 + 986 = 991 \leq 1000$ ✓ et $u_{59} = 5 + 1003 = 1008 > 1000$ ✓
— **Coût comparé** : la boucle effectue 59 additions ; le calcul direct une soustraction et une division ✓
— **Terminaison** : $r = 17 > 0$ donne $u_n \to +\infty$ — arrêt garanti ✓
— **Avantage du calcul direct** : coût **constant**, quel que soit le seuil. Pour un seuil de $10^9$, la boucle ferait 59 millions de tours ✓
— **Contrôle** : $\dfrac{10^9 - 5}{17} \approx 58{,}8$ millions ✓
— **Quand la boucle est nécessaire** : pour une suite définie par récurrence sans forme explicite, comme $u_{n+1} = \sqrt{u_n + 2}$ ✓
— **Somme jusqu'au rang d'arrêt** : $60 \times \dfrac{5 + 1008}{2} = 60 \times 506{,}5 = 30\,390$ ✓

**Réponse.** $n = 59$ ($u_{59} = 1008$) ; la boucle coûte 59 tours, le calcul direct est immédiat.

## Correction de l'exercice 5 – Deux seuils successifs

**Script.**

```
c = 5000.0
```

```
n = 0
while c <= 10000:
    c = 1.04 * c
    n = n + 1
print("10000 :", n)
while c <= 20000:
    c = 1.04 * c
    n = n + 1
print("20000 :", n)
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

— **Premier seuil** : $1{,}04^n > 2$ donne $n > \dfrac{\ln 2}{\ln 1{,}04} \approx \dfrac{0{,}69315}{0{,}039221} \approx 17{,}673$, donc $n = 18$ ✓

— **Vérifications** : $5000 \times 1{,}04^{17} \approx 9739{,}5 \leq 10\,000$ ✓ et $5000 \times 1{,}04^{18} \approx 10\,129{,}1 > 10\,000$ ✓

— **Second seuil** : $1{,}04^n > 4$ donne $n > \dfrac{\ln 4}{\ln 1{,}04} \approx 35{,}346$, donc $n = 36$ ✓

— **Vérifications** : $5000 \times 1{,}04^{35} \approx 19\,730{,}4 \leq 20\,000$ ✓ et $5000 \times 1{,}04^{36} \approx 20\,519{,}7 > 20\,000$ ✓

— **Observation remarquable** : le second doublement demande $36 - 18 = 18$ années, **exactement** comme le premier ✓

— **Explication** : le temps de doublement d'une suite géométrique est **constant**, indépendant du capital de départ ✓

— **Règle des** 70 : $\dfrac{70}{4} = 17{,}5$ ans — très proche des 17,67 exacts ✓

— **Astuce du script** : la seconde boucle **repart** de la valeur courante de $c$ et $n$, sans réinitialiser — c'est exactement ce qu'il faut ✓

— **Erreur à éviter** : réinitialiser $n = 0$ avant la seconde boucle afficherait 18 au lieu de 36 ✓

**Réponse.** 18 ans pour 10 000 € et 36 ans pour 20 000 € (doublement constant de 18 ans).

## Correction de l'exercice 6 – Suite de Syracuse

**Script.**

```
def syracuse(u):
    n = 0
    while u != 1:
        if u % 2 == 0:
            u = u // 2
        else:
            u = 3 * u + 1
        n = n + 1
    return n
```

— **Pour** $u_0 = 6$ : $6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$, soit **huit** étapes ✓

— **Pour** $u_0 = 7$ : $7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$, soit **seize** étapes ✓

— **Altitudes maximales** : 16 pour $u_0 = 6$, et 52 pour $u_0 = 7$ ✓

— **Terminaison** : elle n'est **pas démontrée** ! La conjecture de Syracuse, ouverte depuis 1937, affirme que la suite atteint toujours 1 ✓

— **Vérification informatique** : la conjecture est confirmée pour tous les entiers jusqu'à environ $2^{68}$ ✓

— **Détail technique** : *//* est la division **entière** — indispensable, car */* transformerait $u$ en flottant et le test *u != 1* pourrait ne jamais être satisfait ✓

— **Irrégularité frappante** : $u_0 = 27$ demande 111 étapes et culmine à 9232, alors que $u_0 = 32$ n'en demande que 5 ✓

— **Contrôle** : $32 = 2^5$, donc cinq divisions par 2 ✓

— **Prudence en programmation** : sans preuve de terminaison, il est sage d'ajouter une limite d'itérations.

**Réponse.** 8 étapes pour $u_0 = 6$, 16 pour $u_0 = 7$ (terminaison non démontrée).

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

## Correction de l'exercice 7 – Somme jusqu'à un seuil

**Script.**

```
s = 0
```

```
n = 0
while s <= 500:
    n = n + 1
    s = s + n
print(n, s)
```

— **Résolution exacte** : $\frac{n(n+1)}{2} > 500$ donne $n^2 + n - 1000 > 0$.

— **Discriminant** : $1 + 4000 = 4001$, donc $\sqrt{4001} \approx 63{,}253$ et la racine positive est $\frac{-1 + 63{,}253}{2} \approx 31{,}127$.

— **Réponse** : $n = 32$ ✓

— **Vérifications** : $\frac{31 \times 32}{2} = 496 \leq 500$ ✓ et $\frac{32 \times 33}{2} = 528 > 500$ ✓

— **Affichage** : le script affiche 32 et 528 ✓

— **Ordre des instructions** : on incrémente $n$ **avant** d'ajouter, ce qui garantit que $n$ et $s$ sont cohérents à la sortie ✓

— **Variante à éviter** : ajouter puis incrémenter donnerait $n = 33$ à la sortie, décalé d'une unité ✓

— **Terminaison** : la somme croît sans limite, donc l'arrêt est garanti ✓

— **Coût** : 32 tours — négligeable. Pour un seuil de $10^{12}$, la boucle ferait 1,4 million de tours, là où le calcul direct reste instantané ✓

— **Contrôle** : $\sqrt{2 \times 10^{12}} \approx 1{,}414 \times 10^6$ ✓

**Réponse.** $n = 32$, avec une somme de 528.

## Correction de l'exercice 8 – Dichotomie

**Script.**

```
a, b = 1.0, 2.0
```

```
while b - a > 0.001:
    m = (a + b) / 2
    if m**3 < 5:
        a = m
    else:
        b = m
print(a, b)
```

— **Validité** : $1^3 = 1 < 5$ et $2^3 = 8 > 5$, donc la racine est dans $]1\,;2[$ ✓

— **Nombre d'étapes** : l'amplitude après $k$ étapes vaut $\frac{1}{2^k}$, et $\frac{1}{2^k} \leq 0{,}001$ donne $2^k \geq 1000$.

— **Résolution** : $2^9 = 512$ et $2^{10} = 1024$, donc **dix étapes** ✓

— **Valeur exacte** : $\sqrt[3]{5} \approx 1{,}70998$ ✓

— **Encadrement final** : d'amplitude $\frac{1}{1024} \approx 0{,}000977$, il contient bien 1,70998 ✓

— **Trace des premières étapes** : $m = 1{,}5$ ($3{,}375 < 5$, on garde $[1{,}5\,;2]$) ; $m = 1{,}75$ ($5{,}359 > 5$, on garde $[1{,}5\,;1{,}75]$) ; $m = 1{,}625$ ($4{,}292 < 5$, on garde $[1{,}625\,;1{,}75]$) ✓

— **Terminaison** : l'amplitude est **divisée par** 2 à chaque tour, donc elle tend vers 0 — arrêt garanti ✓

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

— **Efficacité** : 10 étapes pour 3 décimales ; 20 pour 6 décimales — chaque décimale coûte environ 3,32 étapes ✓
— **Contrôle** : $\frac{\ln 10}{\ln 2} \approx 3{,}322$ ✓
— **Comparaison avec Newton** : la méthode de Newton atteindrait $10^{-3}$ en **trois** étapes seulement, mais exige le calcul de la dérivée ✓

**Réponse.** 10 étapes, encadrant $\sqrt[3]{5} \approx 1{,}710$.

## Correction de l'exercice 9 – Recherche d'un diviseur

**Fonction.**

```
def petit_diviseur(n):
    d = 2
    while n % d != 0:
        d = d + 1
    return d
```

— **Pour** 91 : $91 = 7 \times 13$, donc la fonction renvoie 7 ✓
— **Trace** : $d$ vaut 2, 3, 4, 5, 6, puis 7 — six tests ✓
— **Pour** 97 : il est **premier**, donc la fonction renvoie 97 ✓
— **Coût pour** 97 : 96 tests — la boucle va jusqu'à $d = n$ ✓
— **Terminaison** : garantie, car $d = n$ finit toujours par diviser $n$ ✓
— **Optimisation essentielle** : s'arrêter à $\sqrt{n}$, car tout composé a un diviseur $\leq \sqrt{n}$ ✓

```
def petit_diviseur2(n):
    d = 2
    while d * d <= n and n % d != 0:
        d = d + 1
    if d * d > n:
        return n
    return d
```

— **Coût pour** 97 : seulement 9 tests, car $10^2 = 100 > 97$ ✓
— **Gain** : 9 tests au lieu de 96, soit un facteur 10 ✓
— **Pour** $n = 10^{12} + 39$ **(premier)** : la version naïve ferait $10^{12}$ tests (des heures), l'optimisée $10^6$ (instantané) ✓
— **Test de primalité** : $n$ est premier si et seulement si *petit_diviseur2(n)* == $n$ ✓
— **Ordre des conditions** : le *and* teste $d^2 \leq n$ **d'abord** — l'évaluation paresseuse de Python évite ainsi un calcul inutile ✓

**Réponse.** 7 pour 91, 97 pour 97 (premier) ; s'arrêter à $\sqrt{n}$ divise le coût par 10 ici.

## Correction de l'exercice 10 – Seuil sur une suite récurrente

**Script.**

```
from math import sqrt

u = 1.0
n = 0
while u <= 1.99:
    u = sqrt(u + 2)
    n = n + 1
print(n)
```

— **Trace** : $u_1 = \sqrt{3} \approx 1{,}73205$ ; $u_2 \approx 1{,}93185$ ; $u_3 \approx 1{,}98289$ ; $u_4 \approx 1{,}99572$ ✓

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

— **Réponse** : $n = 4$ ✓
— **Point fixe** : $\sqrt{x+2} = x$ donne $x^2 = x + 2$, soit $x^2 - x - 2 = 0$ et $x = 2$ (la racine $-1$ est à rejeter) ✓
— **Limite** : $u_n \to 2$, en croissant et en restant **sous** 2 ✓
— **Terminaison** : le seuil $1{,}99 < 2$ est atteignable, donc la boucle s'arrête ✓
— **Attention** : un seuil de 2,01 ne serait **jamais** atteint — la suite est majorée par 2 ✓
— **Vitesse de convergence** : les écarts à 2 sont 1 ; 0,268 ; 0,068 ; 0,017 ; 0,0043 — ils sont **divisés par environ** 4 à chaque rang ✓
— **Explication** : la dérivée de $x \mapsto \sqrt{x+2}$ en 2 vaut $\dfrac{1}{2\sqrt{4}} = \dfrac{1}{4}$ — c'est le facteur de contraction ✓
— **Absence de forme explicite** : cette suite n'a pas d'expression simple en fonction de $n$, ce qui rend la **boucle indispensable** ✓
— **Rang pour** 1,9999 : il faudrait environ $\dfrac{\ln(10^{-4})}{\ln(1/4)} \approx 6{,}6$, soit $n = 7$ ✓

**Réponse.** $n = 4$ ; terminaison car $u_n \to 2 > 1{,}99$ en croissant.

## Correction de l'exercice 11 – Compteur de chiffres

**Fonction.**

```
def nb_chiffres(n):
    c = 1
    while n >= 10:
        n = n // 10
        c = c + 1
    return c
```

— **Principe** : chaque division entière par 10 **retire** un chiffre ✓
— **Pour** 7 : la boucle ne tourne pas, la fonction renvoie 1 ✓
— **Pour** 1000 : $1000 \to 100 \to 10 \to 1$, soit trois tours, donc 4 chiffres ✓
— **Pour** 123 456 : cinq tours, donc 6 chiffres ✓
— **Terminaison** : $n$ est divisé par 10 à chaque tour, donc il finit par descendre sous 10 ✓
— **Initialisation à** 1 : indispensable, car un nombre à un chiffre ne déclenche aucun tour ✓
— **Lien avec le logarithme** : le nombre de chiffres vaut $\lfloor \log_{10} n \rfloor + 1$ ✓
— **Contrôle** : $\lfloor \log_{10} 123\,456 \rfloor + 1 = \lfloor 5{,}0915 \rfloor + 1 = 6$ ✓
— **Coût** : environ $\log_{10} n$ tours — très économique, 19 tours suffisent pour le plus grand entier sur 64 bits ✓
— **Cas de** $n = 0$ : la fonction renvoie 1, ce qui est le comportement souhaité (« 0 » s'écrit avec un chiffre) ✓
— **Piège de la division** : avec `/` au lieu de `//`, $n$ deviendrait flottant et la boucle tournerait indéfiniment en s'approchant de valeurs comme 1,23456 — sans jamais descendre sous 10 ? Non : $1{,}23456 < 10$, donc elle s'arrêterait, mais avec un compte **faux** ✓

**Réponse.** 1 ; 4 ; 6 chiffres (division entière répétée par 10).

## Correction de l'exercice 12 – Comparer deux suites par boucle

**Script.**

```
n = 1
while 2**n <= n**3:
    n = n + 1
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
print(n)
```

— **Valeurs comparées** :

| $!\,n$ | $2^n$ | $n^3$ |
|---|---|---|
| 8 | 256 | 512 |
| 9 | 512 | 729 |
| 10 | 1024 | 1000 |

— **Réponse** : $n = 10$ ✓
— **Vérifications** : $2^9 = 512 \leq 729$ ✓ et $2^{10} = 1024 > 1000$ ✓
— **Attention au départ** : pour $n = 1$, $2 > 1$ — la condition est **fausse** dès le début et le script afficherait 1 ! ✓
— **Correction** : il faut partir de $n = 2$ (où $4 \leq 8$) pour trouver le vrai franchissement ✓
— **Trace correcte** : de $n = 2$ à $n = 9$, on a $2^n \leq n^3$ ; en $n = 10$, l'inégalité s'inverse ✓
— **Contrôles** : $n = 2$ donne 4 contre 8 ✓ ; $n = 5$ donne 32 contre 125 ✓
— **Comportement à long terme** : l'exponentielle **écrase** définitivement la puissance — au rang 30, $2^{30} \approx 1{,}07 \times 10^9$ contre 27 000 ✓
— **Croissances comparées** : pour tout $k$, $\lim \dfrac{2^n}{n^k} = +\infty$ ✓
— **Leçon sur les boucles** : toujours vérifier la condition **au rang initial** — sinon la boucle peut ne pas tourner du tout.

**Réponse.** $n = 10$ (1024 > 1000), en partant de $n = 2$.

## Correction de l'exercice 13 – Approximation par sommation

**Script.**

```
s = 0
```

```
k = 0
while s <= 1.6:
    k = k + 1
    s = s + 1 / k**2
print(k, s)
```

— **Premiers termes** : 1 ; 1,25 ; 1,3611 ; 1,4236 ; 1,4636 ; 1,4914 ; 1,5118 ; 1,5274 ✓
— **Progression lente** : après 10 termes, $s \approx 1{,}54977$ ; après 20, $\approx 1{,}59616$ ✓
— **Valeurs au voisinage du seuil** : $s_{21} \approx 1{,}598431$ et $s_{22} \approx 1{,}600497$ ✓
— **Réponse** : le script affiche $k = 22$ et $s \approx 1{,}6005$ ✓
— **Limite de la série** : $\dfrac{\pi^2}{6} \approx 1{,}64493$ — résultat célèbre d'Euler ✓
— **Terminaison** : le seuil 1,6 < 1,64493 est atteignable, donc la boucle s'arrête ✓
— **Attention** : un seuil de 1,7 ne serait **jamais** atteint, la série étant majorée par $\dfrac{\pi^2}{6}$ ✓
— **Reste de la série** : $\dfrac{\pi^2}{6} - s_{22} \approx 0{,}0444$, de l'ordre de $\dfrac{1}{22} \approx 0{,}0455$ ✓
— **Rang pour approcher à** $10^{-3}$ : il faudrait environ 1000 termes, le reste étant de l'ordre de $\dfrac{1}{k}$ ✓

**Réponse.** $k = 22$ (avec $s \approx 1{,}6005$) ; la série converge vers $\dfrac{\pi^2}{6} \approx 1{,}645$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

## Correction de l'exercice 14 – Boucle avec double condition

**Script.**

```
n = 0
while n % 7 != 0 or n**2 <= 5000:
    n = n + 1
print(n)
```

— **Condition de continuation** : on avance tant que **l'une** des deux conditions cherchées n'est pas remplie ✓
— **Seuil sur le carré** : $n^2 > 5000$ donne $n > \sqrt{5000} \approx 70{,}711$, donc $n \geq 71$ ✓
— **Premier multiple de** 7 **au-delà** : $7 \times 11 = 77$ ✓
— **Réponse** : $n = 77$ ✓
— **Vérifications** : $77^2 = 5929 > 5000$ ✓ et $77 = 7 \times 11$ ✓
— **Contrôle du candidat précédent** : $70 = 7 \times 10$ est multiple de 7, mais $70^2 = 4900 \leq 5000$ ✓
— **Logique de la condition** : la négation de « A **et** B » est « non A **ou** non B » — d'où le *or* dans la boucle ✓
— **Erreur classique** : écrire *and* au lieu de *or* arrêterait la boucle dès qu'**une seule** condition est satisfaite, donnant $n = 0$ (multiple de 7) ✓
— **Version plus lisible** :

```
n = 71
while n % 7 != 0:
    n = n + 1
print(n)
```

— **Avantage** : on traite le seuil **avant** la boucle, qui ne gère plus que la divisibilité — au plus 7 tours ✓
— **Contrôle** : de 71 à 77, cela fait bien 6 tours ✓

**Réponse.** $n = 77$ ($77^2 = 5929 > 5000$ et $77 = 7 \times 11$).

## Correction de l'exercice 15 – Convergence d'un algorithme

— **Récurrence** : $x_{n+1} = \dfrac{x_n}{2} + \dfrac{1}{x_n}$ — c'est la méthode de Newton pour $x^2 - 2 = 0$ ✓
— **Trace** : $x_1 = 1{,}5$ ; $x_2 \approx 1{,}4166667$ ; $x_3 \approx 1{,}4142157$ ; $x_4 \approx 1{,}4142136$ ✓
— **Écarts à** $\sqrt{2}$ : 0,0858 ; 0,00245 ; $2{,}1 \times 10^{-6}$ ; $1{,}6 \times 10^{-12}$ ✓
— **Critère d'arrêt** : $\left|x^2 - 2\right| \leq 10^{-10}$.
— **Valeurs de** $\left|x^2 - 2\right|$ : 0,25 ; 0,0069 ; $6{,}0 \times 10^{-6}$ ; $4{,}5 \times 10^{-12}$ ✓
— **Réponse** : **quatre** tours ✓
— **Convergence quadratique** : le nombre de décimales correctes **double** à chaque tour — 1, puis 3, puis 6, puis 12 ✓
— **Comparaison avec la dichotomie** : pour $10^{-12}$, il faudrait 40 étapes de dichotomie contre 4 ici ✓
— **Terminaison** : garantie par la convergence quadratique, pourvu que $x_0 > 0$ ✓
— **Danger** : avec $x_0 = 0$, on aurait une **division par zéro** ; avec $x_0 < 0$, la suite convergerait vers $-\sqrt{2}$ et le critère serait tout de même satisfait ✓
— **Limite de précision** : $10^{-10}$ est proche de la précision machine relative ($\approx 2{,}2 \times 10^{-16}$ sur les flottants) — un critère plus exigeant pourrait provoquer une boucle infinie ✓

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

— **Origine historique** : cette formule était déjà utilisée par les Babyloniens, près de 1700 ans avant notre ère.

**Réponse.** Quatre tours, aboutissant à $x \approx 1{,}4142136$.

## Correction de l'exercice 16 – Synthèse : modèle de population

a) **Script.**

```
def population(n):
    p = 12000.0
    for k in range(n):
        p = 0.96 * p + 300
    return p
```

— **Valeurs** : $p_1 = 11\,820$ ; $p_2 = 11\,647{,}2$ ; $p_3 = 11\,481{,}3$ ✓

b) **Limite.** Le point fixe vérifie $0{,}96x + 300 = x$, soit $0{,}04x = 300$ et

$$x = 7500$$

— **Forme explicite** : avec $v_n = p_n - 7500$, on a $v_{n+1} = 0{,}96 v_n$ et $v_0 = 4500$, donc

$$p_n = 7500 + 4500 \times 0{,}96^n$$

— **Contrôles** : $p_0 = 12\,000$ ✓ ; $p_1 = 7500 + 4320 = 11\,820$ ✓ ; $p_3 = 7500 + 3981{,}3 = 11\,481{,}3$ ✓

— **Limite** : $0{,}96^n \to 0$, donc $p_n \to 7500$ individus ✓

c) **Script du seuil.**

```
p = 12000.0
n = 0
while p >= 9000:
    p = 0.96 * p + 300
    n = n + 1
print(n)
```

— **Résolution** : $7500 + 4500 \times 0{,}96^n < 9000$ donne $0{,}96^n < \dfrac{1}{3}$, soit

$$n > \frac{\ln(1/3)}{\ln 0{,}96} \approx \frac{-1{,}0986}{-0{,}040822} \approx 26{,}913$$

— **Réponse** : la $27^{\text{e}}$ année ✓

— **Vérifications** : $p_{26} = 7500 + 4500 \times 0{,}96^{26} \approx 7500 + 1556{,}9 = 9056{,}9 \geq 9000$ ✓

— $p_{27} \approx 7500 + 1494{,}6 = 8994{,}6 < 9000$ ✓

— **Terminaison** : la suite est **décroissante** de limite $7500 < 9000$, donc le seuil est nécessairement franchi ✓

— **Contre-exemple instructif** : un seuil de 7000 ne serait **jamais** atteint, la population étant minorée par 7500 — la boucle tournerait indéfiniment ✓

— **Interprétation démographique** : la population se stabilise là où les 4 % de départs compensent exactement les 300 arrivées : $0{,}04 \times 7500 = 300$ ✓

**Réponse.** a) $p_n = 7500 + 4500 \times 0{,}96^n$ · b) limite 7500 · c) la $27^{\text{e}}$ année (terminaison car $7500 < 9000$).

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).