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 ℓ ; un seuil supérieur à ℓ 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

La boucle double u tant que u reste strictement inférieur à 100, et compte les tours dans n. Après n tours, u vaut donc 2^n, et la boucle s'arrête au premier n tel que $2^n \geq 100$.

On suit les valeurs : u vaut 2, 4, 8, 16, 32, 64, puis 128. Comme

$$2^6 = 64 < 100 \leq 2^7 = 128$$

la boucle s'arrête après le septième tour.

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).

La boucle se termine forcément : la suite (2^n) est géométrique de raison $2 > 1$, elle dépasse tout seuil. Avec *while u <= 100*, le résultat serait le même, car u ne vaut jamais exactement 100. Le script affiche 7, et u vaut alors 128.

Réponse. 7 (avec $u = 128$).

Correction de l'exercice 2 – Boucle de seuil décroissante

On part de $u = 1 = 0{,}95^0$ et l'on multiplie par 0,95 à chaque tour, tant que u reste supérieur ou égal à 0,01. Le compteur n donne alors le premier rang où $0{,}95^n < 0{,}01$.

```
u = 1.0
```

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

La suite $(0{,}95^n)$ est géométrique de raison comprise entre 0 et 1 : elle décroît vers 0, donc elle passe sous tout seuil strictement positif et la boucle se termine. On vérifie le résultat en encadrant :

$$0{,}95^{89} \approx 0{,}01041 \geq 0{,}01 \qquad 0{,}95^{90} \approx 0{,}00989 < 0{,}01$$

Le seuil est franchi de peu : c'est pourquoi on contrôle les deux rangs voisins. Le premier rang cherché est $n = 90$.

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

Correction de l'exercice 3 – Terminaison en défaut

On calcule les premiers termes : u vaut 2, 3, 3,5, 3,75, 3,875… Ils semblent s'approcher de 4. Le point fixe de la relation vérifie $0{,}5x + 2 = x$, soit $x = 4$.

On pose $v_n = u_n - 4$. Alors $v_{n+1} = 0{,}5u_n + 2 - 4 = 0{,}5(u_n - 4) = 0{,}5v_n$: la suite (v_n) est géométrique de raison 0,5 et de premier terme $v_0 = -4$. Donc

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

Comme $4 \times 0{,}5^n > 0$, on a toujours $u_n < 4 < 5$: la condition $u < 5$ reste vraie et le script **ne se termine jamais**.

Avec un seuil atteignable, comme 3,99, la boucle s'arrêterait au rang 9 : $u_8 \approx 3{,}984$ et $u_9 \approx 3{,}992$. **Règle** : avant d'écrire une boucle *while*, on vérifie que le seuil est du bon côté de la limite.

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

On ajoute la raison 17 tant que le terme ne dépasse pas 1000 :

```
u = 5
```

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

Comme la suite est arithmétique, on peut aussi calculer directement : $u_n = 5 + 17n$, et $u_n > 1000$ équivaut à $17n > 995$, soit

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).

$$n > \frac{995}{17} \approx 58{,}5$$

donc $n = 59$. On vérifie : $u_{58} = 991 \leq 1000$ et $u_{59} = 1008 > 1000$.

La boucle fait 59 tours, le calcul direct une seule division. Pour un seuil de 10^9, la boucle ferait près de 59 millions de tours : elle n'est indispensable que pour les suites sans forme explicite. Le premier rang est $n = 59$.

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

Correction de l'exercice 5 – Deux seuils successifs

Placé à 4 %, le capital est multiplié par 1,04 chaque année. On enchaîne deux boucles de seuil ; la seconde **repart** des valeurs courantes de c et n, sans les réinitialiser.

```
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)
```

On contrôle les résultats en encadrant :

$$5000 \times 1{,}04^{17} \approx 9739{,}50 \qquad 5000 \times 1{,}04^{18} \approx 10\,129{,}08$$

puis $5000 \times 1{,}04^{35} \approx 19\,730{,}44$ et $5000 \times 1{,}04^{36} \approx 20\,519{,}66$.

Le second doublement prend autant de temps que le premier, 18 ans : pour une suite géométrique, le temps de doublement ne dépend pas de la valeur de départ. Réinitialiser *n = 0* avant la seconde boucle afficherait 18 au lieu de 36. Réponses : 18 **ans** et 36 **ans**.

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

On applique la règle tant que u est différent de 1, en comptant les étapes. On utilise la division entière *//* pour que u reste un entier.

```
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 8 étapes. Pour $u_0 = 7$, la suite monte jusqu'à 52 avant de redescendre :
$7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$, soit 16 étapes.

Terminaison : personne n'a jamais démontré que cette boucle s'arrête pour tout entier de départ. C'est la conjecture de Syracuse, vérifiée par ordinateur sur d'immenses plages d'entiers mais toujours ouverte. Réponses : 8 **étapes** et 16 **étapes**.

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

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

On ajoute les entiers 1, 2, 3… un par un tant que la somme ne dépasse pas 500. On augmente n **avant** de l'ajouter, pour que n et s correspondent à la sortie de la boucle.

```
s = 0
```

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

On peut le confirmer par le calcul : $1 + \cdots + n = \dfrac{n(n+1)}{2} > 500$ équivaut à $n^2 + n - 1000 > 0$. Le discriminant vaut 4001, et la racine positive est

$$\frac{-1 + \sqrt{4001}}{2} \approx 31{,}1$$

donc $n = 32$. On vérifie : $\dfrac{31 \times 32}{2} = 496 \leq 500$ et $\dfrac{32 \times 33}{2} = 528 > 500$. Le script affiche 32 et 528.

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

Correction de l'exercice 8 – Dichotomie

On part de l'intervalle $[1\,;2]$, qui contient la solution car $1^3 = 1 < 5$ et $2^3 = 8 > 5$. À chaque étape, on calcule le milieu m et l'on garde la moitié qui contient encore la solution : si $m^3 < 5$, la solution est à droite de m.

```
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)
```

Les premières étapes donnent $m = 1{,}5$ (car $3{,}375 < 5$, on garde $[1{,}5\,;2]$), puis $m = 1{,}75$ (car $5{,}359 > 5$, on garde $[1{,}5\,;1{,}75]$). L'amplitude est divisée par 2 à chaque étape : après k étapes, elle vaut $\dfrac{1}{2^k}$. Il faut donc

$$\frac{1}{2^k} \leq 0{,}001 \Leftrightarrow 2^k \geq 1000$$

Comme $2^9 = 512$ et $2^{10} = 1024$, il faut 10 **étapes**. L'encadrement final contient $\sqrt[3]{5} \approx 1{,}710$.

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

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

On teste les diviseurs 2, 3, 4… jusqu'au premier qui divise n, c'est-à-dire jusqu'à ce que le reste $n \,\%\, d$ soit nul. La boucle s'arrête toujours, au pire pour $d = n$.

```
def petit_diviseur(n):
```

```
    d = 2
    while n % d != 0:
```

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).

```
        d = d + 1
    return d
```

Pour $91 = 7 \times 13$, la fonction renvoie 7 après avoir testé 2, 3, 4, 5 et 6. Pour 97, qui est premier, elle teste tous les entiers jusqu'à 97.

On peut s'arrêter bien plus tôt : si n n'est pas premier, il a un diviseur d tel que $d^2 \leq n$. Pour 97, il suffit de tester jusqu'à 9, puisque $10^2 = 100 > 97$:

$$9 \text{ tests au lieu de } 96$$

Il suffit pour cela d'ajouter la condition `d * d <= n` dans la boucle, et de renvoyer n si elle devient fausse. Réponses : 7 pour 91 et 97 pour 97, qui est premier.

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

On calcule les termes avec la fonction *sqrt*, tant que u ne dépasse pas 1,99 :

```
from math import sqrt

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

Les termes valent $u_1 = \sqrt{3} \approx 1{,}7321$, $u_2 \approx 1{,}9319$, $u_3 \approx 1{,}9829$ et $u_4 \approx 1{,}9957$: le script affiche 4.

Terminaison : les termes augmentent et s'approchent de 2, point fixe de la relation, car $\sqrt{x+2} = x$ donne $x^2 - x - 2 = 0$, soit

$$(x-2)(x+1) = 0$$

et seule la solution positive, 2, convient. Le seuil 1,99 est inférieur à 2, donc il est atteint et la boucle s'arrête ; un seuil de 2,01 ne serait jamais atteint. Cette suite n'a pas de forme explicite simple : la boucle est ici indispensable. Le premier rang est $n = 4$.

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

Correction de l'exercice 11 – Compteur de chiffres

Diviser un entier par 10 avec la division entière `//` lui retire son dernier chiffre. On compte donc combien de fois on peut diviser par 10 avant de tomber sous 10, et l'on ajoute 1 pour le chiffre restant : c'est pourquoi le compteur commence à 1.

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

— 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.

La boucle se termine, car n est divisé par 10 à chaque tour. Avec la division décimale `/`, le compte serait faux : 123 456 / 10 donnerait 12 345,6, et ainsi de suite. Réponses : 1, 4 et 6 chiffres.

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

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

On part de $n = 2$, où $2^2 = 4 \leq 2^3 = 8$, et l'on avance tant que $2^n \leq n^3$:

```
n = 2
while 2**n <= n**3:
    n = n + 1
print(n)
```

Partir de $n = 1$ serait une erreur : $2^1 = 2 > 1^3 = 1$, la condition serait fausse dès le départ et le script afficherait 1 sans jamais chercher plus loin.

n	2^n	n^3
8	256	512
9	512	729
10	1024	1000

Au rang 10, l'inégalité s'inverse :

$$2^{10} = 1024 > 10^3 = 1000$$

Par la suite, 2^n distance définitivement n^3 : au rang 30, 2^{30} dépasse le milliard, contre 27 000. Le premier rang cherché est $n = 10$.

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

Correction de l'exercice 13 – Approximation par sommation

On ajoute les termes $\dfrac{1}{k^2}$ un par un tant que la somme ne dépasse pas 1,6 ; comme pour toute somme, on augmente k avant d'ajouter le terme.

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

La somme progresse de plus en plus lentement : 1, 1,25, 1,361, 1,424… puis 1,5962 après 20 termes. On encadre le seuil :

$$s_{21} \approx 1{,}5984 \leq 1{,}6 \qquad s_{22} \approx 1{,}6005 > 1{,}6$$

Le script affiche donc $k = 22$ et $s \approx 1{,}6005$.

Commentaire : la somme ne dépasse jamais $\dfrac{\pi^2}{6} \approx 1{,}645$, un résultat célèbre d'Euler. Un seuil de 1,6 est atteint, mais avec un seuil de 1,7, la boucle ne s'arrêterait jamais.

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

Correction de l'exercice 14 – Boucle avec double condition

On cherche un entier qui vérifie **deux** conditions à la fois. La boucle doit donc continuer tant que l'une au moins n'est pas remplie : la négation de « A et B » est « non A **ou** non B », d'où le *or*.

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).

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

$n^2 > 5000$ impose $n > \sqrt{5000} \approx 70{,}7$, donc $n \geq 71$. Le premier multiple de 7 à partir de 71 est $77 = 7 \times 11$:

$$77^2 = 5929 > 5000$$

Le multiple précédent, 70, ne convient pas, car $70^2 = 4900 \leq 5000$.

Erreur classique : écrire *and* au lieu de *or* arrêterait la boucle dès qu'une seule condition est remplie ; ici dès $n = 0$, qui est multiple de 7. Le script affiche 77.

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

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

Le script applique la relation $x_{n+1} = \dfrac{x_n}{2} + \dfrac{1}{x_n}$ tant que x^2 s'écarte de 2 de plus de 10^{-10}. C'est la méthode de Newton appliquée à l'équation $x^2 - 2 = 0$.

On suit les valeurs : $x_1 = 1{,}5$, $x_2 \approx 1{,}4166667$, $x_3 \approx 1{,}4142157$, $x_4 \approx 1{,}4142136$. Les écarts $\left|x^2 - 2\right|$ correspondants valent environ

$$0{,}25 \qquad 0{,}0069 \qquad 6{,}0 \times 10^{-6} \qquad 4{,}5 \times 10^{-12}$$

Le dernier est inférieur à 10^{-10} : la boucle s'arrête après **quatre** tours.

La méthode est très rapide : le nombre de décimales exactes double à peu près à chaque tour. Une dichotomie demanderait une trentaine d'étapes pour la même précision. Il faut seulement partir d'une valeur strictement positive : avec $x_0 = 0$, on diviserait par zéro. Le script affiche 4 et $x \approx 1{,}4142136$.

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

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

a) Chaque année, la population perd 4 %, donc elle est multipliée par 0,96, puis elle gagne 300 individus :

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

On obtient par exemple $p_1 = 11\,820$ et $p_2 = 11\,647{,}2$.

b) Le point fixe vérifie $0{,}96x + 300 = x$, soit $0{,}04x = 300$ et $x = 7500$. On pose $v_n = p_n - 7500$: alors $v_{n+1} = 0{,}96 v_n$, suite géométrique de premier terme 4500. Donc

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

Comme $0{,}96^n$ tend vers 0, la population tend vers 7500. À ce niveau, les 4 % de départs compensent exactement les 300 arrivées : $0{,}04 \times 7500 = 300$.

c) Même boucle avec *while p >= 9000* et un compteur. La population décroît vers $7500 < 9000$, donc elle passe forcément sous 9000 et la boucle se termine. On encadre : $p_{26} \approx 9056{,}9 \geq 9000$ et $p_{27} \approx 8994{,}6 < 9000$. C'est la 27^{e} **année**.

Réponse. a) $p_n = 7500 + 4500 \times 0{,}96^n$ · b) limite 7500 · c) la 27^{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).