Plaquette 2 – Conditions, boucles while et seuils

Algorithmique et Python — Terminale spé maths

Corrections détaillées

Boîte à outils

Test : *if … elif … else* exécute le premier bloc dont la condition est vraie. Opérateurs : == (égalité), *!*= (différent), <=, >=, *and*, *or*, *not* ; *n % d* est le reste de la division de *n* par *d*.

Boucle de seuil : pour trouver le premier rang n tel que $u_n > A$,

```
n = 0
u = u0
while u <= A:      # contraire de l'objectif
    u = f(u)
    n = n + 1
print(n)
```

Terminaison : la boucle s'arrête si la condition finit par devenir fausse. Il faut le **justifier** (par exemple $u_n \to +\infty$). Sinon, la boucle est **infinie**.

Vérification par le calcul : un seuil sur une suite géométrique se retrouve avec le logarithme,

$$q^n > A \Leftrightarrow n > \frac{\ln A}{\ln q} \quad (q > 1)$$

Contrôle : on vérifie toujours la valeur **juste avant** et **au** rang trouvé.

Correction de l'exercice 1 – Une fonction avec des conditions

Python teste les conditions **dans l'ordre** et s'arrête à la première vraie :

— 17 : 17 ≥ 16, renvoie « TB » ;

— 14 : le premier test échoue, le deuxième (14 ≥ 14) réussit : « B » ;

— 11,5 : seul le test ≥ 10 réussit : « Passable » ;

— 9 : aucun test ne réussit : « Refusé ».

Ordre des tests : si l'on testait d'abord *note >= 10*, une note de 17 recevrait « Passable ». Il faut tester les seuils du **plus grand au plus petit**, chaque *elif* supposant que les tests précédents ont échoué :

$$\text{« B »} \Leftrightarrow 14 \leq \text{note} < 16$$

Réponse. « TB », « B », « Passable », « Refusé ».

Correction de l'exercice 2 – Compter avec une condition

```
c = 0
for n in range(1, 100):
    if n % 3 == 0 or n % 5 == 0:
        c = c + 1
print(c)
```

Le script affiche 46.

Vérification (principe d'inclusion-exclusion) :

— multiples de 3 : 33 ; multiples de 5 : 19 ;
— multiples de 15 (comptés deux fois) : 6.

$$33 + 19 - 6 = 46 \ \checkmark$$

— **Opérateur** : *n % 3 == 0* teste si le reste de la division par 3 est nul, c'est-à-dire si 3 divise n.

Réponse. 46.

Correction de l'exercice 3 – Dépasser un million

```
n = 0
```

```
p = 1
while p <= 10**6:
    p = 2 * p
    n = n + 1
print(n)
```

Le script affiche 20 (et p vaut alors $2^{20} = 1\,048\,576$).

Vérification :

$$2^n > 10^6 \Leftrightarrow n \ln 2 > 6 \ln 10 \Leftrightarrow n > \frac{6 \ln 10}{\ln 2} \approx 19{,}93$$

— Le premier entier est 20 ✓ ; en effet $2^{19} = 524\,288 \leq 10^6$.
— **Terminaison** : $2^n \to +\infty$, donc la condition de la boucle finit par être fausse.

Réponse. $n = 20$.

Correction de l'exercice 4 – Un seuil vers la limite

a) La condition de la boucle est le **contraire** de l'objectif $u_n < 11$:

```
n = 0
```

```
u = 50
while u >= 11:
    u = 0.9 * u + 1
    n = n + 1
print(n)
```

Le script affiche 36.

b) $u_n < 11 \Leftrightarrow 40 \times 0{,}9^n < 1 \Leftrightarrow 0{,}9^n < \frac{1}{40}$, et $\ln 0{,}9 < 0$:

$$n > \frac{\ln(1/40)}{\ln 0{,}9} = \frac{\ln 40}{-\ln 0{,}9} \approx 35{,}01$$

— Premier entier : 36 ✓
— **Terminaison** : $u_n \to 10 < 11$, donc la suite finit par passer sous 11.

Réponse. $n = 36$.

Correction de l'exercice 5 – Une somme qui dépasse 5

On ajoute les termes un par un tant que la somme ne dépasse pas 5 :

```
s = 0
```

```
n = 0
while s <= 5:
    n = n + 1
    s = s + 1/n
```

```
print(n)
```

Le script affiche 83 (la somme vaut alors environ 5,002).

— **Terminaison** : on admet que la somme $H_n = \sum_{k=1}^{n} \frac{1}{k}$ tend vers $+\infty$, bien que ses termes tendent vers 0 :

$$H_n \approx \ln n + 0{,}577$$

— **Contrôle** : $\ln 83 + 0{,}577 \approx 4{,}42 + 0{,}58 = 5{,}0$ ✓

— **Remarque** : la croissance est très lente ; pour dépasser 20, il faudrait environ 270 millions de termes.

Réponse. $n = 83$.

Correction de l’exercice 6 – Une boucle qui ne s’arrête pas

La suite est croissante et **converge vers le point fixe** :

$$\ell = 0{,}5\,\ell + 3 \Leftrightarrow \ell = 6$$

— On a $u_n = 6 - 6 \times 0{,}5^n < 6$ pour tout n : la suite ne dépasse **jamais** 6, et encore moins 10.

— La condition $u < 10$ reste toujours vraie : la boucle est **infinie**.

Leçon : avant d’écrire une boucle de seuil, il faut s’assurer **mathématiquement** que le seuil est atteint (ici, il faudrait un seuil strictement inférieur à 6, par exemple 5,9).

— **Garde-fou** : on peut ajouter une condition de sécurité, par exemple *while u < 10 and n < 1000*.

Réponse. La suite converge vers $6 < 10$: la condition reste toujours vraie.

Correction de l’exercice 7 – Années bissextiles

On traduit la règle par une condition composée :

```
def bissextile(a):
    return (a % 4 == 0 and a % 100 != 0) or a % 400 == 0
```

— 2024 : divisible par 4, pas par 100 : **bissextile**.

— 1900 : divisible par 100 mais pas par 400 : **non bissextile**.

— 2000 : divisible par 400 : **bissextile**.

$$\text{bissextile} \Leftrightarrow (4 \mid a \text{ et } 100 \nmid a) \text{ ou } 400 \mid a$$

— **Contrôle** : sur 400 ans, il y a $100 - 4 + 1 = 97$ années bissextiles, d’où une année moyenne de $365 + \frac{97}{400} = 365{,}2425$ jours.

Réponse. 2024 oui, 1900 non, 2000 oui.

Correction de l’exercice 8 – Le temps de doublement

On part d’un indice 1 et on multiplie par 1,02 tant qu’on n’a pas atteint 2 :

```
P = 1
```

```
n = 0
while P < 2:
    P = P * 1.02
    n = n + 1
print(n)
```

Le script affiche 36.

Vérification :

$$1{,}02^n \geq 2 \Leftrightarrow n \geq \frac{\ln 2}{\ln 1{,}02} \approx 35{,}003$$

— Le premier entier qui convient est donc 36 : en effet $1{,}02^{35} \approx 1{,}999\,9$, **juste** sous 2, et $1{,}02^{36} \approx 2{,}040$.
— **Piège** : arrondir 35,003 à 35 ferait répondre 35 à tort ; le seuil n'est franchi qu'au cours de la 36e année.
— **Règle de 70** : $\frac{70}{2} = 35$ donne le bon ordre de grandeur.

Réponse. 36 ans.

Correction de l'exercice 9 – Nombres parfaits

Pour chaque n, on additionne les diviseurs $d < n$, repérés par la condition $n \bmod d = 0$:

```
for n in range(2, 1000):
    s = 0
    for d in range(1, n):
        if n % d == 0:
            s = s + d
    if s == n:
        print(n)
```

Le script affiche 6, 28 et 496.

— **Vérification pour** 28 : diviseurs $1, 2, 4, 7, 14$, de somme

$$1 + 2 + 4 + 7 + 14 = 28 \ \checkmark$$

— **Coût** : environ $\frac{1\,000^2}{2} = 500\,000$ divisions ; on pourrait se limiter aux diviseurs jusqu'à $\sqrt{n}$.

Réponse. 6, 28 et 496.

Correction de l'exercice 10 – Approcher une limite à 0,001 près

```
n = 1
while abs(3 + 2/n - 3) >= 0.001:
    n = n + 1
print(n)
```

Le script affiche 2 001.

Vérification : $|u_n - 3| = \frac{2}{n}$, et

$$\frac{2}{n} < 10^{-3} \Leftrightarrow n > 2\,000$$

— Le premier entier est 2 001 ✓ ($n = 2\,000$ donne exactement 10^{-3}, qui n'est pas **strictement** inférieur).
— **Lien** : c'est la traduction algorithmique de la définition de la limite, « à partir d'un certain rang, u_n est aussi proche que voulu de 3 ».

Réponse. $n = 2\,001$.

Correction de l'exercice 11 – L'algorithme d'Euclide

a) Chaque tour remplace $(a\,;b)$ par $(b\,;r)$, où r est le reste de a par b :

a	b	**reste** $a \bmod b$
1 071	462	147
462	147	21
147	21	0
21	0	fin

La fonction renvoie 21 = PGCD(1 071 ; 462).

b) Le reste vérifie $0 \leq r < b$: la valeur de b **diminue strictement** à chaque tour tout en restant un entier positif.

$$b_0 > b_1 > b_2 > \cdots \geq 0$$

Une suite d'entiers positifs strictement décroissante ne peut pas être infinie : b finit par valoir 0.

Réponse. a) 21 · b) b décroît strictement dans $\mathbb{N}$.

Correction de l'exercice 12 – Factorielle et milliard

On met à jour la factorielle à chaque tour, $n! = n \times (n-1)!$:

```
n = 1
f = 1
while f <= 10**9:
    n = n + 1
    f = f * n
print(n)
```

Le script affiche 13.

Vérification :

$$12! = 479\,001\,600 \leq 10^9 \qquad 13! = 6\,227\,020\,800 > 10^9$$

— **Remarque** : mettre à jour f à partir de sa valeur précédente évite de recalculer toute la factorielle à chaque tour (un seul produit par tour).

Réponse. $n = 13$.

Correction de l'exercice 13 – Deux conditions d'arrêt

La boucle s'arrête dès que **l'une** des deux conditions devient fausse : $u \geq 100$ **ou** $n \geq 50$.

Formule : après n tours, u vaut $1 + 0{,}5\,(0 + 1 + \cdots + (n-1))$:

$$u = 1 + 0{,}5 \times \frac{n(n-1)}{2} = 1 + \frac{n(n-1)}{4}$$

— $n = 20$: $u = 1 + 95 = 96 < 100$; $n = 21$: $u = 1 + 105 = 106 \geq 100$.

— La boucle s'arrête avec $n = 21$ et $u = 106$ (la condition $n < 50$ n'a pas servi).

— **Intérêt** : la seconde condition est un **garde-fou** qui garantit l'arrêt même si le seuil n'était jamais atteint.

Réponse. $n = 21$ et $u = 106$.

Correction de l'exercice 14 – Chercher un minimum par balayage

On garde en mémoire la **meilleure** valeur trouvée et on la met à jour quand on trouve plus petit :

```
x = 0
```

```
meilleur = 0
xmin = 0
while x <= 5:
    v = x**2 - 4*x
    if v < meilleur:
        meilleur = v
        xmin = x
    x = round(x + 0.01, 2)
print(xmin, meilleur)
```

Le script affiche 2.0 et -4.0.

Vérification : forme canonique $f(x) = (x-2)^2 - 4$, minimum -4 en $x = 2$.

$$f'(x) = 2x - 4 = 0 \Leftrightarrow x = 2$$

— **Détail technique** : *round(x + 0.01, 2)* évite l'accumulation d'erreurs d'arrondi des nombres à virgule.

Réponse. Minimum -4 en $x = 2$.

Correction de l'exercice 15 – Concentration d'un médicament

a) Script :

```
n = 1
```

```
C = 10
while C <= 30:
    C = 0.7 * C + 10
    n = n + 1
print(n)
```

Le script affiche 7 : c'est après la 7**e dose** que la concentration dépasse 30 mg/L ($C_7 \approx 30{,}6$).

b) Point fixe :

$$\ell = 0{,}7\,\ell + 10 \Leftrightarrow \ell = \frac{10}{0{,}3} \approx 33{,}3$$

— La concentration tend vers environ 33,3 mg/L (« état d'équilibre » du traitement).

— **Terminaison de a)** : C_n croît vers 33,3 > 30, donc le seuil est atteint.

Réponse. a) après la 7e dose · b) $\frac{100}{3} \approx 33{,}3$ mg/L.