

Plaquette 4 – Seuils, algorithmes et tableur

Suites numériques et modèles — Terminale STMG

Corrections détaillées

Boîte à outils

La boucle « tant que ». Elle répète l'évolution **tant que** le seuil n'est pas franchi, et compte les tours.

```
def seuil():
    u = 2000
    n = 0
    while u < 4000:
        u = u * 1.06
        n = n + 1
    return n
```

Élément	Rôle
la variable de valeur	elle porte u_n
le compteur	il porte le rang n
la condition	elle décrit ce qui reste à faire

△ La condition est la **négarion** de l'objectif : pour « dépasser 4000 », on boucle tant que $u < 4000$.

△ L'ordre compte : on met à jour la valeur **puis** le compteur, sinon le rang renvoyé est faux.

Renvoyer la valeur plutôt que le rang. On change simplement ce que renvoie la fonction.

La boucle « pour » pour un cumul.

```
def cumul(n):
    u = 1200
    s = 0
    for k in range(n):
        s = s + u
        u = u + 150
    return s
```

Au tableur. On écrit le premier terme, puis une formule de récurrence que l'on recopie vers le bas.

Modèle	Formule en B3
arithmétique, $r = 150$	$= B2 + 150$
géométrique, $q = 1,06$	$= B2 \times 1,06$
mixte $0,9u + 500$	$= B2 \times 0,9 + 500$

Correction de l'exercice 1 – Compléter un algorithme de seuil

- **Objectif** : dépasser 4000 ✓
- **Condition de boucle** : tant que l'objectif n'est **pas** atteint ✓

- **Écriture** : $u < 4000$ ✓
- **Coefficient de croissance** : 1,06 ✓
- **Algorithme complété** :

```
def seuil():
    u = 2000
    n = 0
    while u < 4000:
        u = u * 1.06
        n = n + 1
    return n
```

- **Premier tour** : u vaut 2120, n vaut 1 ✓
- **Deuxième tour** : $u \approx 2247,20$, n vaut 2 ✓
- **Après 11 tours** : $u \approx 3796,60$ ✓
- **La condition est encore vraie** ✓
- **Après 12 tours** : $u \approx 4024,39$ ✓
- **La condition devient fausse** ✓
- **Valeur renvoyée** :

12

- **Contrôle par la formule** : $1,06^{12} \approx 2,0122$ ✓
- **Concordance** ✓
- **Piège sur la condition** : écrire $u > 4000$ ✓
- **Effet** : la boucle ne démarre jamais ✓
- **Valeur renvoyée alors** : 0 ✓

Réponse. *while* $u < 4000$ et $u = u \mid 1.06^*$; la fonction renvoie 12.

Correction de l'exercice 2 – Un seuil vers le bas

- **Objectif** : passer **sous** 1000 ✓
- **Condition de boucle** : tant que le stock reste au-dessus ✓
- **Écriture** : $s > 1000$ ✓
- **Coefficient** : 0,85 ✓
- **Algorithme** :

```
def seuil():
    s = 5000
    n = 0
    while s > 1000:
        s = s * 0.85
        n = n + 1
    return n
```

- **Premier tour** : $s = 4250$, $n = 1$ ✓
- **Après 9 tours** : $s \approx 1158,08$ ✓
- **Condition encore vraie** ✓
- **Après 10 tours** : $s \approx 984,37$ ✓
- **Condition fausse** ✓
- **Valeur renvoyée** :

10

- **Remarque sur le sens de la condition** : il s'inverse ✓

- **Raison** : la suite est décroissante ✓
- **Erreur classique** : garder $s < 1000$ ✓
- **Effet** : la boucle ne s'exécute pas ✓
- **Contrôle par la formule** : $0,85^{10} \approx 0,1969$ ✓
- **Produit** : $\approx 984,37$ ✓
- **Concordance** ✓

Réponse. `while s > 1000` avec $s = s \setminus 0.85^*$; la fonction renvoie 10.

Correction de l'exercice 3 - Trace d'exécution

- **Valeur initiale** : 100 ✓
- **Coefficient** : 1,15 ✓
- **Rang 1** : $100 \times 1,15$ ✓
- **Valeur** : 115 ✓
- **Rang 2** : $115 \times 1,15$ ✓
- **Valeur** : 132,25 ✓
- **Rang 3** : $132,25 \times 1,15$ ✓
- **Valeur** : $\approx 152,09$ ✓
- **Rang 4** : $\approx 174,90$ ✓
- **Rang 5** : $\approx 201,14$ ✓
- **Tableau** :

n	0	1	2	3	4	5
u_n	100	115	132,25	152,09	174,90	201,14

- **Contrôle par la formule** : $100 \times 1,15^5$ ✓
- **Calcul de $1,15^5$** : $\approx 2,01136$ ✓
- **Valeur** : $\approx 201,14$ ✓
- **Concordance** ✓
- **Rang où l'on dépasse 200** :

$$n = 5$$

- **Contrôle au rang 4** : $174,90 < 200$ ✓
- **Attention aux arrondis** : ne jamais réutiliser une valeur arrondie ✓
- **Bonne pratique** : garder la valeur exacte en mémoire ✓

Réponse. 100 ; 115 ; 132,25 ; 152,09 ; 174,90 ; 201,14.

Correction de l'exercice 4 - Formules de tableur

a) **Premier terme** : 1200 ✓

— **Formule** :

$$= 1200$$

— **Remarque** : on peut aussi saisir la valeur directement ✓

b) **Modèle** : on ajoute 150 ✓

— **Formule** :

$$= B2 + 150$$

— **Recopie vers le bas** : la référence B2 devient B3, puis B4... ✓

— **Vocabulaire** : c'est une référence **relative** ✓

— **Contrôle en B3** : 1350 ✓

— **Contrôle en B4** : 1500 ✓

c) **Terme général** : $p_n = 150n + 1200$ ✓

— **Formule** :

$$= 150 \times A2 + 1200$$

— **Contrôle pour A2 = 0** : 1200 ✓

— **Contrôle pour A10 = 8** : 2400 ✓

— **Avantage de b)** : elle marche pour tout modèle de récurrence ✓

— **Avantage de c)** : elle donne un terme isolé sans les précédents ✓

— **Cas où seule b) marche** : un modèle mixte ✓

— **Exemple** : $u_{n+1} = 0,9u_n + 500$ ✓

— **Formule correspondante** : $= B2 \times 0,9 + 500$ ✓

— **Piège de recopie** : figer une cellule avec des dollars quand il ne faut pas ✓

Réponse. a) = 1200 · b) = $B2 + 150$ · c) = $150 \times A2 + 1200$.

Correction de l'exercice 5 - Tableur et modèle géométrique

a) **Coefficient** : 1,06 ✓

— **Formule** :

$$= B2 \times 1,06$$

— **Contrôle** : 2120 ✓

b) **Terme général** : $2000 \times 1,06^n$ ✓

— **Formule** : $=2000 \backslash 1,06^{\wedge} A3$ (l'accent circonflexe note la puissance)

— **Symbole de la puissance** : l'accent circonflexe ✓

— **Contrôle pour A3 = 1** : 2120 ✓

c) **Référence au taux** : la cellule D1 ✓

— **Formule naïve** : $= B2 \times (1 + D1)$ ✓

— **Problème à la recopie** : D1 deviendrait D2 ✓

— **Solution** : figer la cellule ✓

— **Écriture** : $=B2 \backslash (1+D1)$ en plaçant un signe dollar devant la lettre et** devant le numéro de D1

— **Vocabulaire** : une référence **absolue** ✓

— **Rôle des dollars** : bloquer la colonne et la ligne ✓

— **Intérêt** : changer D1 met tout le tableau à jour ✓

— **Test avec 8 % en D1** : B3 affiche 2160 ✓

— **Test avec 3 %** : B3 affiche 2060 ✓

— **Bonne pratique** : sortir les paramètres du modèle dans des cellules dédiées ✓

— **Avantage** : on ne réécrit jamais les formules ✓

Réponse. a) $=B2 \backslash 1,06 \cdot$ b) $=2000 \backslash 1,06^{\wedge} A3 \cdot$ c) $=B2 \backslash (1+D1)$ ** avec D1 figée par des dollars.

Correction de l'exercice 6 - Que renvoie cet algorithme ?

— **Modèle** : arithmétique de raison 60 ✓

— **Valeur initiale** : 500 ✓

- **Condition** : tant que $u < 800$ ✓
- **Tour 1** : $u = 560, n = 1$ ✓
- **Tour 2** : $u = 620, n = 2$ ✓
- **Tour 3** : $u = 680, n = 3$ ✓
- **Tour 4** : $u = 740, n = 4$ ✓
- **Tour 5** : $u = 800, n = 5$ ✓
- **Test de la condition** : $800 < 800$ est **faux** ✓
- **Sortie de la boucle** ✓
- **Valeur renvoyée** :

5

- **Contrôle par le calcul** : $500 + 60n \geq 800$ ✓
- **Inéquation** : $60n \geq 300$ ✓
- **Solution** : $n \geq 5$ ✓
- **Concordance** ✓
- **Si la condition était $u \leq 800$ ** :* un tour de plus ✓
- **Valeur renvoyée alors** : 6 ✓
- **Différence de sens** : « atteindre » contre « dépasser strictement » ✓
- **Leçon** : lire attentivement l'inégalité ✓

Réponse. 5.

Correction de l'exercice 7 - Modifier un algorithme

- **Pour renvoyer le capital** : changer la dernière ligne ✓
- **Écriture** : `return u` ✓
- **Algorithme** :

```
def capital():
    u = 2000
    n = 0
    while u < 4000:
        u = u * 1.06
        n = n + 1
    return u
```

- **Valeur renvoyée** : $\approx 4024,39$ ✓
- **Pour renvoyer les deux** : `return n, u` ✓
- **Résultat** : le couple $(12; 4024,39)$ ✓
- **Vocabulaire** : Python renvoie un **tuple** ✓
- **Utilité** : une seule exécution donne les deux réponses ✓
- **Autre variante utile** : afficher chaque étape ✓
- **Instruction** : `print(n, u)` dans la boucle ✓
- **Effet** : on voit la trace complète ✓
- **Variante paramétrée** : passer le seuil en argument ✓
- **En-tête** : `def capital(seuil):` ✓
- **Condition** : `while u < seuil` ✓
- **Appel** : `capital(5000)` renvoie $\approx 5080,70$ ✓
- **Rang correspondant** : 16 ✓
- **Appel** : `capital(3000)` renvoie $\approx 3007,26$ ✓
- **Rang** : 7 ✓

— **Avantage** : une seule fonction pour tous les seuils ✓

Réponse. *return u* donne $\approx 4024,39$; *return n, u* donne le couple.

Correction de l'exercice 8 - Un cumul par boucle

— **Deux variables** : la production courante et le cumul ✓

— **Initialisation du cumul** : 0 ✓

— **Algorithme** :

```
def cumul(n):
    u = 1200
    s = 0
    for k in range(n):
        s = s + u
        u = u + 150
    return s
```

— **Ordre des deux lignes** : ajouter **avant** d'incrémenter ✓

— **Raison** : le premier mois vaut 1200 ✓

— **Exécution pour $n = 3$** : $1200 + 1350 + 1500$ ✓

— **Valeur** : 4050 ✓

— **Pour $n = 20$** : dernier terme $1200 + 150 \times 19$ ✓

— **Valeur** : 2850 ✓

— **Moyenne** : $\frac{1200 + 2850}{2}$ ✓

— **Valeur** : 2025 ✓

— **Cumul** : 20×2025 ✓

— **Valeur** :

52 500

— **Contrôle pour $n = 19$** : $19 \times \frac{1200 + 3900}{2}$ ✓

— **Valeur** : 48 450 ✓

— **Différence** : 4050, le 20^e terme ✓

— **Concordance** ✓

— **Piège** : inverser les deux lignes de la boucle ✓

— **Effet** : on oublierait le premier terme ✓

Réponse. Le cumul vaut 52 500 unités sur 20 mois.

Correction de l'exercice 9 - Un seuil sur un cumul

— **Deux compteurs** : le cumul et le rang ✓

— **Condition** : tant que le cumul reste sous 50 000 ✓

— **Algorithme** :

```
def seuil_cumul():
    u = 1200
    s = 0
    n = 0
    while s < 50000:
        s = s + u
        u = u + 150
        n = n + 1
    return n
```

- **Cumul après 19 mois** : 48 450 ✓
- **Condition encore vraie** ✓
- **Cumul après 20 mois** : 52 500 ✓
- **Condition fausse** ✓
- **Valeur renvoyée** :
20 mois
- **Contrôle** : $48\,450 < 50\,000 < 52\,500$ ✓
- **Concordance** ✓
- **Production du 20^e mois** : 2850 ✓
- **Dépassement** : 2500 unités ✓
- **Différence avec un seuil sur u_n** : deux questions distinctes ✓
- **Seuil $u_n > 3000$** : au 13^e rang ✓
- **Ici on cumule, là on regarde un terme isolé** ✓
- **Piège** : oublier d'initialiser s à 0 ✓
- **Effet** : une erreur d'exécution ✓

Réponse. 20 mois, pour un cumul de 52 500 unités.

Correction de l'exercice 10 - Corriger un algorithme

- **Ce que renvoie la version actuelle** : la valeur u ✓
- **Valeur** : $\approx 4024,39$ ✓
- **Correction demandée** : *return n* ✓
- **Valeur alors renvoyée** : 12 ✓
- **Remarque sur l'ordre des deux lignes** : ici il est sans effet ✓
- **Raison** : les deux s'exécutent à chaque tour complet ✓
- **Ce qui compte** : qu'elles soient toutes deux **dans** la boucle ✓
- **Cas où l'ordre compterait** : si l'on affichait u et n ensemble ✓
- **Algorithme corrigé** :

```
def seuil():
    u = 2000
    n = 0
    while u < 4000:
        u = u * 1.06
        n = n + 1
    return n
```

- **Erreur bien plus grave** : sortir l'incrément de la boucle ✓
- **Effet** : n vaudrait 1 ✓
- **Autre erreur grave** : oublier la mise à jour de u ✓
- **Effet** : une boucle **infinie** ✓
- **Signe d'alerte** : le programme ne s'arrête pas ✓
- **Vérification systématique** : la variable testée doit évoluer ✓
- **Ici** : u est bien multipliée à chaque tour ✓
- **Et elle croît** : la condition finira par être fausse ✓

Réponse. Elle renvoie $\approx 4024,39$; il faut écrire *return n*, qui donne 12.

Correction de l'exercice 11 - Un modèle mixte au tableur

- a) **Traduction directe du modèle** ✓

— **Formule :**

$$= B2 \times 0,9 + 500$$

— **Recopie vers le bas ✓**

b) **Calcul de u_1 :** $0,9 \times 8000 + 500$ ✓

— **Produit :** 7200 ✓

— **Valeur :**

$$u_1 = 7700$$

— **Calcul de u_2 :** $0,9 \times 7700 + 500$ ✓

— **Produit :** 6930 ✓

— **Valeur :** $u_2 = 7430$ ✓

— **Calcul de u_3 :** $0,9 \times 7430 + 500$ ✓

— **Produit :** 6687 ✓

— **Valeur :** $u_3 = 7187$ ✓

c) **Observation :** la suite décroît ✓

— **Mais de moins en moins vite ✓**

— **Baisses successives :** 300 ; 270 ; 243 ✓

— **Ces baisses sont géométriques de raison 0,9 ✓**

— **Point d'équilibre :** $x = 0,9x + 500$ ✓

— **Résolution :** $0,1x = 500$ ✓

— **Valeur :**

$$x = 5000$$

— **Interprétation :** la base tend vers 5000 clients ✓

— **Contrôle :** si $u_n = 5000$, alors $u_{n+1} = 5000$ ✓

— **Valeur au rang 8 :** $\approx 6291,40$ ✓

— **Lecture commerciale :** 500 nouveaux clients compensent 10 % de pertes à 5000 ✓

Réponse. a) $= B2 \times 0,9 + 500$ · b) 7700 ; 7430 ; 7187 · c) la base tend vers 5000 clients.

Correction de l'exercice 12 - Une épargne annuelle

— **Algorithme :**

```
def annees():
    u = 0
    n = 0
    while u < 100000:
        u = u * 1.03 + 15000
        n = n + 1
    return n
```

— **Année 1 :** $u = 15\,000$ ✓

— **Année 2 :** $15\,000 \times 1,03 + 15\,000$ ✓

— **Valeur :** 30450 ✓

— **Année 3 :** $\approx 46\,363,50$ ✓

— **Année 4 :** $\approx 62\,754,41$ ✓

— **Année 5 :** $\approx 79\,637,04$ ✓

— **Année 6 :** $\approx 97\,026,15$ ✓

— **Condition encore vraie ✓**

- **Année 7** : $\approx 114\,936,93$ ✓
- **Condition fausse** ✓
- **Valeur renvoyée** :
7 ans
- **Total versé** : $7 \times 15\,000$ ✓
- **Valeur** : $105\,000$ € ✓
- **Intérêts gagnés** : $\approx 9936,93$ € ✓
- **Sans intérêts** : le seuil serait franchi en 7 ans aussi ✓
- **Détail** : $6 \times 15\,000 = 90\,000$ € ✓
- **Les intérêts ne changent pas le rang ici** ✓
- **Mais ils changent le montant** : ≈ 9937 € de plus ✓

Réponse. 7 ans, pour $\approx 114\,937$ € accumulés.

Correction de l'exercice 13 - Un parc à renouveler

- **Coefficient** : 0,88 ✓
- **Condition** : tant que le parc dépasse 6000 ✓
- **Algorithme** :

```
def renouvellement():
    p = 12000
    n = 0
    while p > 6000:
        p = p * 0.88
        n = n + 1
    return n
```

- **Année 1** : 10 560 ✓
- **Année 2** : $\approx 9292,80$ ✓
- **Année 3** : $\approx 8177,66$ ✓
- **Année 4** : $\approx 7196,34$ ✓
- **Année 5** : $\approx 6332,78$ ✓
- **Condition encore vraie** ✓
- **Année 6** : $\approx 5572,85$ ✓
- **Condition fausse** ✓
- **Valeur renvoyée** :

6 ans

- **Contrôle par la formule** : $0,88^6 \approx 0,4644$ ✓
- **Produit** : $\approx 5572,85$ ✓
- **Concordance** ✓
- **Contrôle au rang 5** : $0,88^5 \approx 0,5277$ ✓
- **Produit** : $\approx 6332,78$ ✓
- **Supérieur à 6000** ✓
- **Conclusion confirmée** ✓

Réponse. 6 ans, le parc tombant à ≈ 5573 machines.

Correction de l'exercice 14 - Comparer deux algorithmes

- a) **Algorithme A** : il multiplie 10 fois par 1,04 ✓

- **Résultat** : $1000 \times 1,04^{10}$ ✓
- **Calcul de** $1,04^{10}$: $\approx 1,48024$ ✓
- **Valeur** :
 $\approx 1480,24$
- **Algorithme B** : la même chose par la formule ✓
- **Valeur** :
 $\approx 1480,24$
- **Conclusion** : ils renvoient la même valeur ✓
- b) **Avantage de B** : plus court et plus rapide ✓
- **Autre avantage** : pas d'accumulation d'arrondis ✓
- **Avantage de A** : on peut afficher les étapes ✓
- **Autre avantage de A** : il s'adapte à un modèle mixte ✓
- **Exemple où B ne marche pas** : $u_{n+1} = 0,9u_n + 500$ ✓
- **Raison** : aucune formule simple du terme général en STMG ✓
- **Règle pratique** : formule si elle existe, boucle sinon ✓
- **Pour un seuil** : la boucle *while* s'impose ✓
- **Raison** : on ne connaît pas le nombre de tours à l'avance ✓
- **Pour un rang fixé** : la boucle *for* ou la formule ✓
- **Contrôle croisé** : faire les deux et comparer ✓
- **Ici** : les deux donnent $\approx 1480,24$ ✓
- **Concordance** ✓

Réponse. a) tous deux $\approx 1480,24$ · b) B pour un rang fixé, A dès que le modèle n'a pas de formule simple.

Correction de l'exercice 15 - Repérer une erreur d'algorithme

- **Variable testée** : u ✓
- **Est-elle modifiée dans la boucle ?** Non ✓
- **Conséquence** : la condition reste vraie ✓
- **Comportement** :
boucle infinie
- **Signe visible** : le programme ne rend jamais la main ✓
- **Ligne manquante** : la mise à jour de u ✓
- **Écriture** : $u = u \mid 0.85*$ ✓
- **Algorithme corrigé** :

```
def seuil():
    u = 5000
    n = 0
    while u > 1000:
        u = u * 0.85
        n = n + 1
    return n
```

- **Valeur renvoyée** : 10 ✓
- **Règle de vérification n°1** : la variable testée doit évoluer ✓
- **Règle n°2** : elle doit évoluer **dans le bon sens** ✓
- **Contre-exemple** : $u = u \mid 1.15*$ avec la même condition ✓

- **Effet** : u croît, la condition reste vraie ✓
- **Encore une boucle infinie** ✓
- **Règle n°3** : le seuil doit être atteignable ✓
- **Contre-exemple** : viser $u \leq 0$ avec un modèle géométrique positif ✓
- **Effet** : jamais atteint ✓
- **Bonne pratique** : tester d’abord sur quelques tours à la main ✓

Réponse. Boucle infinie : il manque $u = u \setminus 0.85*$ dans la boucle ; corrigée, elle renvoie 10.

Correction de l’exercice 16 - Synthèse : un plan d’investissement

a) **Calcul de u_1** : $1,05 \times 40\,000 - 3000$ ✓

— **Produit** : 42 000 ✓

— **Valeur** :

$$u_1 = 39\,000 \text{ €}$$

— **Calcul de u_2** : $1,05 \times 39\,000 - 3000$ ✓

— **Produit** : 40 950 ✓

— **Valeur** : $u_2 = 37\,950 \text{ €}$ ✓

— **Calcul de u_3** : $1,05 \times 37\,950 - 3000$ ✓

— **Produit** : 39 847,50 ✓

— **Valeur** : $u_3 = 36\,847,50 \text{ €}$ ✓

— **Observation** : le fonds diminue ✓

b) **Condition** : tant que le fonds reste positif ✓

— **Algorithme** :

```
def epuisement():
```

```
    u = 40000
```

```
    n = 0
```

```
    while u > 0:
```

```
        u = u * 1.05 - 3000
```

```
        n = n + 1
```

```
    return n
```

— **Valeur renvoyée** : 23 ✓

— **Contrôle au rang 22** : $\approx 1494,79 \text{ €}$ ✓

— **Contrôle au rang 23** : $\approx -1430,48 \text{ €}$ ✓

— **Conclusion** : le fonds est épuisé la 23^e année ✓

c) **Équation** : $x = 1,05x - 3000$ ✓

— **Isolement** : $3000 = 0,05x$ ✓

— **Valeur** :

$$x = 60\,000 \text{ €}$$

— **Interprétation** : à 60 000 €, le fonds serait stable ✓

— **Ici** : $40\,000 < 60\,000$ ✓

— **Conséquence** : le fonds s’épuise ✓

Réponse. a) 39 000 ; 37 950 ; 36 847,50 € · b) l’algorithme renvoie 23 · c) 60 000 €.