

Plaquette 4 – Seuils, algorithmes et tableur

Suites numériques – Première STI2D

Corrections détaillées

Boîte à outils

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

```
def seuil():
    u = 450
    n = 0
    while u < 900:
        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

Pour une suite géométrique croissante, la boucle s'arrête au premier rang n tel que

$$u_0 \times q^n \geq S$$

△ La condition est la **négarion** de l'objectif. Pour « descendre sous 1500 », on boucle tant que la valeur **dépasse** 1500.

La boucle « pour » pour un cumul.

```
def cumul(n):
    u = 100
    s = 0
    for k in range(n):
        s = s + u
        u = u * 1.08
    return s
```

Au tableur.

Modèle	Formule en B3
arithmétique, $r = 24$	$= B2 + 24$
géométrique, $q = 1,06$	$= B2 * 1,06$
mixte $0,85u + 3$	$= B2 * 0,85 + 3$

On recopie vers le bas : les références **relatives** se décalent toutes seules.

Trois erreurs à traquer.

Erreur	Symptôme
la variable testée n'évolue pas	boucle infinie
l'incrément est hors de la boucle	rang renvoyé faux
le seuil est inatteignable	boucle infinie

Correction de l'exercice 1 - Compléter un algorithme croissant

On veut **dépasser** 900 : la boucle doit continuer **tant que** la production reste inférieure à 900, d'où la condition $u < 900$. Une hausse de 6 % revient à multiplier par 1,06 : $u = u * 1.06$.

```
def seuil():
    u = 450
    n = 0
    while u < 900:
        u = u * 1.06
        n = n + 1
    return n
```

Après 11 tours, $u \approx 854,23$: la condition est encore vraie. Après 12 tours, $u \approx 905,49$: elle devient fausse et la fonction renvoie

$$n = 12$$

Contrôle : $1,06^{12} \approx 2,012$. Écrire $u > 900$ serait une erreur : la condition serait fausse dès le départ, et la fonction renverrait 0.

Réponse. $while\ u < 900$ et $u = u * 1.06$; la fonction renvoie 12.

Correction de l'exercice 2 - Un algorithme décroissant

On veut passer **sous** 1 500 W : on continue tant que la consommation reste au-dessus, d'où la condition $c > 1500$. Baisser de 8 % revient à multiplier par 0,92.

```
def seuil():
    c = 2400
    n = 0
    while c > 1500:
        c = c * 0.92
        n = n + 1
    return n
```

Après 5 tours, $c \approx 1\,581,80$ W, encore au-dessus ; après 6 tours, $c \approx 1\,455,25$ W. La fonction renvoie 6.

$$2\,400 \times 0,92^6 \approx 1\,455,25$$

Pour une suite décroissante, le sens de la condition s'inverse par rapport à une suite croissante ; garder $c < 1500$ empêcherait la boucle de démarrer.

Réponse. $while\ c > 1500$ avec $c = c * 0.92$; la fonction renvoie 6.

Correction de l'exercice 3 - Trace d'exécution

On applique la relation de récurrence de proche en proche, en gardant les valeurs exactes en mémoire :

$$T_1 = 0,85 \times 85 + 3 = 72,25 + 3 = 75,25$$

puis $T_2 = 0,85 \times 75,25 + 3 \approx 66,96$, $T_3 \approx 59,92$, $T_4 \approx 53,93$ et $T_5 \approx 48,84$ °C.

n	0	1	2	3	4	5
T_n	85	75,25	66,96	59,92	53,93	48,84

Les baisses successives ($-9,75$; $-8,29$; $-7,04$; $-5,99$; $-5,09$) diminuent : le refroidissement ralentit. La suite n'est ni arithmétique (différences non constantes) ni géométrique.

Réponse. 85 ; 75,25 ; 66,96 ; 59,92 ; 53,93 ; 48,84 °C.

Correction de l'exercice 4 - Formules de tableur

Une cellule de la colonne B contient la valeur du rang précédent (B2) transformée par la relation de récurrence.

- a) Suite arithmétique de raison 24 : on ajoute 24, d'où $=B2+24$ (B3 vaut alors 344). On peut aussi utiliser la formule explicite $=24*A3+320$.
- b) Suite géométrique de raison 1,06 : on multiplie, d'où $=B2*1,06$ (B3 vaut 477), ou directement $=450*1,06^A3$.
- c) Le modèle mixte se traduit par $=B2*0,85+3$ (B3 vaut 75,25). Il n'a pas de formule explicite au programme : seule la récurrence est utilisable.

$$T_{n+1} = 0,85 T_n + 3 \rightarrow = B2 * 0,85 + 3$$

En recopiant vers le bas, la référence relative B2 devient B3, B4... Pour placer le taux dans une cellule D1, on écrit $=B2*(1+D1)$ en figeant la référence D1 (référence absolue), afin qu'elle ne change pas lors de la recopie.

Réponse. a) $= B2 + 24$ · b) $= B2 * 1,06$ · c) $= B2 * 0,85 + 3$.

Correction de l'exercice 5 - Que renvoie cet algorithme ?

La boucle ajoute 24 à u tant que $u < 500$, en comptant les tours. On suit les valeurs de u : 344, 368, 392, 416, 440, 464, 488, puis 512 au huitième tour. Après ce tour, le test $512 < 500$ est faux : la boucle s'arrête et la fonction renvoie

$$n = 8$$

Contrôle par le calcul : on cherche le premier n tel que $320 + 24n \geq 500$, soit $24n \geq 180$ et $n \geq 7,5$, donc $n = 8$. Avec la condition $u \leq 500$, le résultat serait le même, car aucun terme ne vaut exactement 500.

Réponse. 8.

Correction de l'exercice 6 - Modifier un algorithme

Pour renvoyer la production plutôt que le rang, on remplace la dernière ligne par *return* u : la fonction renvoie alors $450 \times 1,06^{12} \approx 905,49$. Avec *return* n , u , elle renverrait le couple (12 ; 905,49).

Pour choisir le seuil à l'appel, on le passe en argument :

```
def production(seuil):
    u = 450
    n = 0
    while u < seuil:
        u = u * 1.06
        n = n + 1
    return n
```

L'appel *production(1200)* renvoie 17, car

$$450 \times 1,06^{16} \approx 1\,143,16 < 1\,200 < 450 \times 1,06^{17} \approx 1\,211,75$$

et *production(700)* renvoie 8. Une seule fonction sert pour tous les seuils.

Réponse. *return u* donne $\approx 905,49$; avec un argument, *production(1200)* renvoie 17.

Correction de l'exercice 7 – Un cumul par boucle

Il faut deux variables : la production du mois, *u*, et le cumul, *s*, qui vaut 0 au départ. À chaque tour, on ajoute la production du mois au cumul, **puis** on la fait augmenter de 8 % :

```
def cumul(n):
    u = 100
    s = 0
    for k in range(n):
        s = s + u
        u = u * 1.08
    return s
```

Pour $n = 3$, la fonction calcule $100 + 108 + 116,64 = 324,64$. Pour $n = 10$, elle additionne les dix productions mensuelles, de 100 à $100 \times 1,08^9 \approx 199,90$:

$$100 + 108 + 116,64 + \dots + 199,90 \approx 1\,448,66 \text{ pièces}$$

Si l'on inversait les deux lignes de la boucle, le premier mois compterait 108 pièces au lieu de 100, et le cumul vaudrait environ 1 564,55.

Réponse. Le cumul vaut $\approx 1448,66$ pièces sur 10 mois.

Correction de l'exercice 8 – Un seuil sur un cumul

Il faut trois variables : la production *u*, le cumul *s* et le nombre de mois *n*. On continue tant que le cumul n'a pas atteint 2 000 :

```
def seuil_cumul():
    u = 100
    s = 0
    n = 0
    while s < 2000:
        s = s + u
        u = u * 1.08
        n = n + 1
    return n
```

Après 12 mois, le cumul vaut environ 1 897,71 pièces, encore insuffisant ; après 13 mois, environ 2 149,53. La fonction renvoie

$$n = 13$$

Attention à tester *s* et non *u* : le seuil porte sur le cumul. La production mensuelle, elle, dépasse 200 pièces dès le rang 10.

Réponse. 13 mois, pour un cumul de $\approx 2149,53$ pièces.

Correction de l'exercice 9 – Corriger un algorithme

La condition porte sur *c*, mais *c* n'est jamais modifiée dans la boucle : la condition $c > 1\,500$ reste vraie indéfiniment et le programme ne s'arrête jamais (boucle infinie). Il manque la mise à jour $c = c * 0.92$:

```
def seuil():
```

```

c = 2400
n = 0
while c > 1500:
    c = c * 0.92
    n = n + 1
return n

```

Corrigée, la fonction renvoie 6. Trois règles à vérifier avant de lancer une boucle *while* :
la variable testée évolue, dans le bon sens, vers un seuil atteignable

Par exemple, $c = c * 1.08$ avec la même condition donnerait encore une boucle infinie, puisque c augmenterait.

Réponse. Boucle infinie : il manque $c = c * 0.92$; corrigée, elle renvoie 6.

Correction de l'exercice 10 - Un modèle mixte au tableur

a) On traduit directement la relation : $=B2*0,95+200$, puis on recopie vers le bas.

b) On calcule de proche en proche :

$$v_1 = 0,95 \times 500 + 200 = 675 \quad v_2 = 0,95 \times 675 + 200 = 841,25$$

et $v_3 = 0,95 \times 841,25 + 200 \approx 999,19$ L.

c) La cuve se remplit, mais de moins en moins vite : les gains valent 175, puis 166,25, puis environ 157,94 L. Au tableur, on observe que le volume continue de monter vers 4 000 L sans le dépasser (environ 1 678 L au rang 8). On peut vérifier que 4 000 L est un niveau stable : $0,95 \times 4\,000 + 200 = 4\,000$; à ce niveau, la fuite de 5 %, soit 200 L, compense exactement l'apport.

Réponse. a) $= B2 * 0,95 + 200$ · b) 675 ; 841,25 ; $\approx 999,19$ L · c) le volume tend vers 4000 L.

Correction de l'exercice 11 - Un seuil sur un modèle mixte

Le modèle n'a pas de formule explicite au programme : on le fait tourner jusqu'à dépasser 3 000 L.

```

def remplissage():
    v = 500
    n = 0
    while v < 3000:
        v = v * 0.95 + 200
        n = n + 1
    return n

```

Les valeurs successives donnent environ 1 904 L au rang 10, 2 745 L au rang 20, 2 978 L au rang 24, puis

$$v_{25} \approx 3\,029,14 \text{ L}$$

La fonction renvoie 25. Viser 4 000 L ferait tourner la boucle indéfiniment, car le volume s'en approche sans jamais l'atteindre ; un seuil de 3 800 L serait atteint au rang 56.

Réponse. L'algorithme renvoie 25 ($\approx 3\,029,14$ L).

Correction de l'exercice 12 - Repérer une erreur

L'indentation délimite les blocs en Python : la ligne $n = n + 1$ est au même niveau que *while*, donc **hors** de la boucle. La boucle tourne bien 12 fois (u atteint environ 905,49), mais le compteur n'augmente qu'une seule fois, après la sortie. La fonction renvoie

$$n = 1$$

au lieu de 12. Pour corriger, on décale la ligne $n = n + 1$ au même niveau que $u = u * 1.06$, à l'intérieur de la boucle ; la fonction renvoie alors 12.

Pour détecter ce genre d'erreur, on peut afficher les variables à chaque tour avec `print(n, u)`, ou comparer à un calcul fait à la main.

Réponse. Il renvoie 1 : la ligne $n = n + 1$ est hors de la boucle.

Correction de l'exercice 13 - Comparer deux approches

a) L'algorithme A multiplie 12 fois 450 par 1,06 ; l'algorithme B applique directement la formule explicite. Ils renvoient la même valeur :

$$450 \times 1,06^{12} \approx 905,49$$

b) B est plus court et n'accumule pas d'arrondis : on le préfère quand on connaît la formule explicite et le rang. A permet d'afficher les étapes et s'adapte aux modèles sans formule explicite, comme $v_{n+1} = 0,95 v_n + 200$. Pour chercher un seuil, dont le nombre d'étapes est inconnu à l'avance, on utilise une boucle `while`. Faire les deux calculs et les comparer est un bon contrôle.

Réponse. a) tous deux $\approx 905,49$ · b) B pour un rang fixé, A dès que le modèle est mixte ou qu'on veut la trace.

Correction de l'exercice 14 - Un refroidissement à surveiller

On fait tourner le modèle tant que la température reste au-dessus de 30 °C :

```
def refroidissement():
    T = 85
    n = 0
    while T > 30:
        T = T * 0.85 + 3
        n = n + 1
    return n
```

Les températures successives valent environ 48,84 °C au rang 5, 37,71 °C au rang 8, 32,80 °C au rang 10, 30,88 °C au rang 11, puis

$$T_{12} \approx 29,25 \text{ °C}$$

La fonction renvoie 12. La température se rapproche de 20 °C, celle du local, sans l'atteindre : un seuil de 20 °C ferait tourner la boucle sans fin, alors qu'un seuil de 25 °C est franchi au rang 16 ($T_{16} \approx 24,83 \text{ °C}$).

Réponse. L'algorithme renvoie 12 ($\approx 29,25 \text{ °C}$).

Correction de l'exercice 15 - Deux boucles à comparer

Une boucle `for k in range(10)` fait exactement 10 tours : on l'utilise quand le nombre d'étapes est **connu**, par exemple pour calculer u_{10} ou cumuler sur 10 périodes. Elle s'arrête toujours.

Une boucle `while u < 900` tourne **tant que** la condition est vraie : on l'utilise quand le nombre d'étapes est **inconnu**, par exemple pour chercher le nombre de mois nécessaires pour dépasser 900. Elle risque de ne jamais s'arrêter si la condition reste vraie.

for:rang fixé *while*:seuil à franchir

On peut écrire un `while` avec un compteur (`while k < 10`), ou borner une recherche avec `for k in range(1000)` et un test de sortie pour éviter le blocage.

Réponse. *for* : nombre de tours connu (calculer u_{10}) ; *while* : nombre inconnu (chercher

un seuil).

Correction de l'exercice 16 - Synthèse : un banc de charge

a) On applique la relation pas à pas : $u_1 = 0,7 \times 0 + 3,6 = 3,60$ V,
 $u_2 = 0,7 \times 3,6 + 3,6 = 6,12$ V et $u_3 = 0,7 \times 6,12 + 3,6 \approx 7,88$ V.

b) Si la tension vaut 12 V à une étape, elle vaut à l'étape suivante

$$0,7 \times 12 + 3,6 = 8,4 + 3,6 = 12 \text{ V}$$

elle ne bouge plus : 12 V est la tension du générateur, dont la charge se rapproche sans l'atteindre.

c) On fait tourner le modèle tant que $u < 10$:

```
def charge():  
    u = 0  
    n = 0  
    while u < 10:  
        u = u * 0.7 + 3.6  
        n = n + 1  
    return n
```

On obtient environ 9,12 V au rang 4 et 9,98 V au rang 5 (de peu sous 10 V), puis 10,59 V au rang 6 : la fonction renvoie 6. Viser 12 V provoquerait une boucle infinie.

Réponse. a) 3,60 ; 6,12 ; $\approx 7,88$ V · b) $0,7 \times 12 + 3,6 = 12$ · c) l'algorithme renvoie 6.