

## Plaquette 3 – Suites et seuils : boucles bornées et non bornées

### Algorithmique et Python — Terminale STI2D

Corrections détaillées

#### Boîte à outils

##### Suites et algorithmes.

- **Suite arithmétique** de raison  $r$  :  $u_{n+1} = u_n + r$ , en Python  $u = u + r$ . Terme général :  $u_n = u_0 + nr$ .
- **Suite géométrique** de raison  $q$  :  $u_{n+1} = u_n \times q$ , en Python  $u = u * q$ . Terme général :  $u_n = u_0 \times q^n$ .
- **Évolution de  $t\%$**  : la raison vaut  $q = 1 + \frac{t}{100}$  — donc 1,04 pour +4 % et 0,85 pour -15 %.
- **Suite arithmético-géométrique** :  $u_{n+1} = au_n + b$ , en Python  $u = a * u + b$ . Sa limite éventuelle  $L$  vérifie

$$L = aL + b \quad \text{soit} \quad L = \frac{b}{1-a} \quad (a \neq 1)$$

- **Calculer  $u_n$  (nombre de tours connu)** : boucle *for*  $i$  in  $\text{range}(n)$  : qui applique  $n$  fois la relation de récurrence à partir de  $u_0$ .
- **Chercher un seuil (nombre de tours inconnu)** : boucle *while*, avec un compteur.

```
def seuil():
    u = 5000
    n = 0
    while u <= 8000:
        u = u * 1.035
        n = n + 1
    return n
```

- **Somme de termes** : une variable accumulateur  $s = s + u$  **avant** de faire évoluer  $u$ , si l'on veut la somme de  $u_0$  à  $u_{n-1}$ .
- **Somme d'une suite géométrique** (contrôle du résultat) :

$$u_0 + u_1 + \dots + u_{n-1} = u_0 \times \frac{1 - q^n}{1 - q} \quad (q \neq 1)$$

- **Contrôle obligatoire** : tout résultat de boucle doit être retrouvé par la formule explicite. Un programme juste et un calcul juste doivent concorder.

#### Correction de l'exercice 1 - Une suite arithmétique

La boucle ajoute 18 exactement  $n$  fois à la valeur initiale 250 : c'est la suite arithmétique de premier terme  $u_0 = 250$  et de raison  $r = 18$ .

- Pour  $n = 12$ , la boucle tourne 12 fois et ajoute  $12 \times 18 = 216$ .
- La fonction renvoie  $250 + 216 = 466$ .

##### Contrôle par la formule explicite.

$$u_n = u_0 + nr$$

- $u_{12} = 250 + 12 \times 18 = 466$ . Les deux méthodes concordent.

- **Ce qu’il faut voir** :  $u = u + 18$  est la traduction Python de  $u_{n+1} = u_n + 18$ . Reconnaître la nature de la suite dans le code permet de contrôler le résultat sans dérouler la boucle.

**Réponse.** 466.

### Correction de l’exercice 2 - Une suite géométrique

- a) Une hausse de 4 % correspond à une multiplication par 1,04 : c’est une suite géométrique de raison  $q = 1,04$ .

```
def capital(n):
    u = 1500
    for i in range(n):
        u = u * 1.04
    return u
```

- b) On applique la formule explicite pour contrôler.

$$u_n = u_0 \times q^n = 1\,500 \times 1,04^n$$

- $1,04^{10} \approx 1,480\,244$ .
- $u_{10} \approx 1\,500 \times 1,480\,244 = 2\,220,366\,4$ , soit 2 220,37 €.
- **Piège fréquent** : écrire  $u = u + 0.04$  au lieu de  $u = u * 1.04$ . La première ligne ajoute 4 centimes par an, la seconde augmente de 4 % — deux modèles totalement différents.

**Réponse.** b) environ 2 220,37 €.

### Correction de l’exercice 3 - Une suite définie par récurrence

- a) La relation se traduit ligne à ligne.

```
def u(n):
    valeur = 200
    for i in range(n):
        valeur = 0.8 * valeur + 30
    return valeur
```

- b) On déroule la relation.

| $n$   | 0   | 1   | 2   | 3     | 4      | 5       |
|-------|-----|-----|-----|-------|--------|---------|
| $u_n$ | 200 | 190 | 182 | 175,6 | 170,48 | 166,384 |

- $u_1 = 0,8 \times 200 + 30 = 160 + 30 = 190$ .
- $u_2 = 0,8 \times 190 + 30 = 152 + 30 = 182$ .
- En poursuivant :  $u_5 = 166,384$ .
- **Attention à l’ordre des opérations** : il faut multiplier **puis** ajouter. Écrire  $0.8 * (valeur + 30)$  donnerait une tout autre suite.

**Réponse.** b)  $u_1 = 190$ ,  $u_2 = 182$ ,  $u_5 = 166,384$ .

### Correction de l’exercice 4 - Franchir un seuil vers le haut

Le nombre d’années est **inconnu** : c’est donc une boucle *while*.

```
def seuil():
    u = 5000
    n = 0
    while u <= 8000:
        u = u * 1.035
```

```
n = n + 1
return n
```

**Contrôle par le calcul.** On cherche le plus petit  $n$  tel que  $5\,000 \times 1,035^n > 8\,000$ .

| $n$                     | <b>13</b> | <b>14</b> |
|-------------------------|-----------|-----------|
| $5\,000 \times 1,035^n$ | 7 819,78  | 8 093,47  |

- Pour  $n = 13$ , le capital vaut 7 819,78 € : encore insuffisant.
- Pour  $n = 14$ , il vaut 8 093,47 € : le seuil est franchi.
- La fonction renvoie 14 : il faut 14 ans.
- **Rédaction attendue** : toujours montrer que  $n - 1$  ne convient **pas**. Donner seulement la valeur qui convient ne prouve pas qu'elle est la plus petite.

**Réponse.** 14 ans.

### Correction de l'exercice 5 - Franchir un seuil vers le bas

Une baisse de 15 % correspond à une multiplication par  $1 - 0,15 = 0,85$ .

```
def decote():
    v = 12000
    n = 0
    while v >= 4000:
        v = v * 0.85
        n = n + 1
    return n
```

**Contrôle par le calcul.**

| $n$                     | <b>6</b> | <b>7</b> |
|-------------------------|----------|----------|
| $12\,000 \times 0,85^n$ | 4 525,79 | 3 846,93 |

- Pour  $n = 6$  : 4 525,79 €  $\geq$  4 000 €, la boucle continue.
- Pour  $n = 7$  : 3 846,93 €  $<$  4 000 €, la boucle s'arrête.
- La fonction renvoie 7 : au bout de 7 ans.
- **Attention au sens de la condition** : pour une descente, la boucle continue tant que la valeur est **au-dessus** du seuil. Écrire `while v <= 4000` : ne ferait aucun tour, puisque 12 000 est déjà supérieur à 4 000.

**Réponse.** 7 ans.

### Correction de l'exercice 6 - Accumuler une somme

L'accumulateur  $s$  ajoute la valeur **avant** que  $v$  ne soit modifié : la fonction additionne donc  $v_0, v_1, \dots, v_9$ , soit les dix premiers versements.

**Contrôle par la formule du cours** (somme des  $n$  premiers termes d'une suite géométrique) :

$$S = v_0 \times \frac{1 - q^n}{1 - q} = 1\,200 \times \frac{1 - 1,025^{10}}{1 - 1,025}$$

- $1,025^{10} \approx 1,280\,085$ .
- $S \approx 1\,200 \times \frac{-0,280\,085}{-0,025} = 1\,200 \times 11,203\,38 \approx 13\,444,06$ .
- La fonction renvoie environ 13 444,06 €.

— **Ordre des deux lignes** : si  $v = v * 1.025$  venait avant  $s = s + v$ , on additionnerait  $v_1$  à  $v_{10}$  et on obtiendrait 13 780,16 €. Placer l'accumulation en premier n'a rien d'anodin.

**Réponse.** Environ 13 444,06 €.

### Correction de l'exercice 7 - Deux placements qui se croisent

Le nombre de tours est inconnu : boucle *while*, avec **deux** suites qui évoluent en parallèle.

```
def croisement():
    a = 4000
    b = 4000
    n = 0
    while b <= a:
        a = a + 300
        b = b * 1.06
        n = n + 1
    return n
```

**Contrôle par le calcul.**

| $n$ | 7        | 8        | 9        |
|-----|----------|----------|----------|
| A   | 6 100    | 6 400    | 6 700    |
| B   | 6 014,52 | 6 375,39 | 6 757,92 |

- Pour  $n = 8$ , B vaut encore moins que A.
- Pour  $n = 9$ , B dépasse A pour la première fois.
- La fonction renvoie 9 : c'est au bout de 9 ans.
- **Interprétation** : une croissance de 6 % démarre plus lentement qu'un ajout fixe de 300 € — 6 % de 4 000 € ne font que 240 € — mais elle finit toujours par l'emporter, car elle porte sur un capital qui grossit.

**Réponse.** Au bout de 9 ans.

### Correction de l'exercice 8 - Afficher un tableau de valeurs

L'instruction *print* affiche  $n$  et la valeur **courante** de  $u$ , avant la division.

| ligne affichée | $n$ | $u$ |
|----------------|-----|-----|
| 1              | 0   | 64  |
| 2              | 1   | 32  |
| 3              | 2   | 16  |
| 4              | 3   | 8   |

- Quatre lignes sont affichées : 0 64, 1 32, 2 16, 3 8.
- Après le dernier affichage,  $u$  vaut encore 4, mais cette valeur n'est **jamais** affichée : la boucle s'arrête avant.
- **Piège** : déplacer *print* après la division afficherait 32, 16, 8, 4. La position de l'affichage dans la boucle change complètement la sortie.

**Réponse.** 0 64, 1 32, 2 16, 3 8.

## Correction de l'exercice 9 - Rembourser un emprunt

a) Chaque mois, le capital restant est d'abord majoré des intérêts, puis diminué de la mensualité.

$$c_{n+1} = c_n \times 1,004 - 450$$

$$- c_1 = 15\,000 \times 1,004 - 450 = 15\,060 - 450 = 14\,610 \text{ €}.$$

$$- c_2 = 14\,610 \times 1,004 - 450 = 14\,668,44 - 450 = 14\,218,44 \text{ €}.$$

b) Le nombre de mensualités est inconnu : boucle *while*, qui tourne tant que le capital reste positif.

```
def duree():
    c = 15000
    n = 0
    while c > 0:
        c = c * 1.004 - 450
        n = n + 1
    return n
```

— Au 35<sup>e</sup> mois, il reste 379,62 € ; au 36<sup>e</sup>, le capital devient négatif.

— La fonction renvoie 36 : l'emprunt est soldé en 36 mensualités, la dernière étant réduite.

— **Contrôle de vraisemblance** :  $36 \times 450 = 16\,200$  € versés pour 15 000 € empruntés, soit 1 200 € d'intérêts. C'est cohérent avec un taux de 0,4 % par mois sur trois ans.

**Réponse.** b) 36 mensualités.

## Correction de l'exercice 10 - Vers une valeur limite

a) Si la suite se stabilise en  $L$ , alors  $u_{n+1}$  et  $u_n$  valent tous deux  $L$  :

$$L = 0,8L + 30$$

$$- L - 0,8L = 30, \text{ soit } 0,2L = 30.$$

$$- L = \frac{30}{0,2} = 150.$$

— La suite décroît de 200 vers 150 sans jamais l'atteindre.

b) Le seuil 155 est au-dessus de la limite : la boucle s'arrêtera.

```
def rang():
    u = 200
    n = 0
    while u >= 155:
        u = 0.8 * u + 30
        n = n + 1
    return n
```

— Valeurs :  $u_5 \approx 166,38$ ,  $u_6 \approx 163,11$ ,  $u_7 \approx 160,49$ ,  $u_8 \approx 158,39$ ,  $u_9 \approx 156,71$ ,  $u_{10} \approx 155,37$ ,  $u_{11} \approx 154,29$ .

— La fonction renvoie 11.

— **Avertissement important** : ce programme ne fonctionne que parce que le seuil 155 est **au-dessus** de la limite 150. Avec un seuil de 149, la boucle serait infinie : la suite s'en approcherait sans jamais le franchir.

**Réponse.** a)  $L = 150$  · b) rang 11.

## Correction de l'exercice 11 - Doubler un capital

```
def doublement():
    u = 1
    n = 0
    while u < 2:
        u = u * 1.07
        n = n + 1
    return n
```

**Contrôle par le calcul.** On cherche le plus petit  $n$  tel que  $1,07^n \geq 2$ .

| $n$      | 10        | 11        |
|----------|-----------|-----------|
| $1,07^n$ | 1,967 151 | 2,104 852 |

- $n = 10$  ne suffit pas ;  $n = 11$  convient.
- La réponse est 11 **ans**.

**Indépendance vis-à-vis du capital.** Un capital  $C$  devient  $C \times 1,07^n$ .

- La condition « avoir doublé » s'écrit  $C \times 1,07^n \geq 2C$ , et  $C$  se simplifie des deux côtés.
- Il reste  $1,07^n \geq 2$  : le capital initial n'intervient pas. C'est pourquoi le programme peut partir de  $u = 1$ .
- **Repère utile** : un taux de 7 % double un capital en environ 10 ans, un taux de 3,5 % en environ 20 ans. Le produit « taux  $\times$  durée » reste voisin de 70.

**Réponse.** 11 ans, indépendamment du capital initial.

## Correction de l'exercice 12 - Choisir la bonne boucle

a) Le nombre de tours est **donné** par l'énoncé : 8 trimestres.

- On utilise une boucle **bornée** : `for i in range(8):`.
- La réponse est la valeur finale de la variable.

b) Le nombre de tours est précisément **ce qu'on cherche**.

- On utilise une boucle **non bornée** : `while stock >= 500:`, avec un compteur  $n = n + 1$ .
- La réponse est la valeur finale du compteur, pas celle du stock.
- **Le critère de choix** : la question commence-t-elle par « quelle valeur » (boucle `for`) ou par « au bout de combien » (boucle `while`) ? Toutes les questions de seuil du programme se reconnaissent à cette formulation.
- *\*Précaution propre au while\** : vérifier avant d'écrire que la suite franchit bien le seuil, sinon la boucle ne s'arrête jamais.

**Réponse.** a) boucle `for` (nombre de tours connu) · b) boucle `while` avec compteur (nombre de tours cherché).

## Correction de l'exercice 13 - Deux populations

```
def depassement():
    a = 8000
    b = 5000
    n = 0
    while b <= a:
        a = a * 0.98
        b = b * 1.04
        n = n + 1
```

```
return n
```

### Contrôle par le calcul.

| $n$ | 7       | 8       |
|-----|---------|---------|
| A   | 6 945,0 | 6 806,1 |
| B   | 6 579,7 | 6 842,9 |

- Après 7 ans, A domine encore ; après 8 ans, B passe devant.
- La fonction renvoie 8 : c'est au bout de 8 ans.
- **Remarque de modélisation** : ces modèles ne valent que sur une durée limitée. Prolongé sur un siècle, le modèle de B donnerait plus de 250 000 habitants, ce qui n'a aucun sens pour une petite ville. Toujours discuter le domaine de validité.

**Réponse.** Au bout de 8 ans.

### Correction de l'exercice 14 - Concentration d'un médicament

a) On élimine d'abord 30 %, c'est-à-dire qu'il reste 70 %, puis on ajoute 40 mg.

$$d_{n+1} = 0,7 d_n + 40 \quad \text{avec} \quad d_0 = 100$$

b) On déroule, comme le ferait le programme  $d = 0.7 * d + 40$ .

| $n$        | 0   | 1   | 2   | 3     |
|------------|-----|-----|-----|-------|
| $d_n$ (mg) | 100 | 110 | 117 | 121,9 |

- $d_1 = 0,7 \times 100 + 40 = 70 + 40 = 110$ .
- $d_2 = 0,7 \times 110 + 40 = 77 + 40 = 117$ .
- $d_3 = 0,7 \times 117 + 40 = 81,9 + 40 = 121,9$ .
- c) La valeur limite  $L$  vérifie  $L = 0,7L + 40$ .
- $0,3L = 40$ , donc  $L = \frac{40}{0,3} \approx 133,33$  mg.
- La concentration monte et se stabilise autour de 133 mg.
- **Lecture médicale** : c'est le « plateau » du traitement. Il ne dépend que de la dose et du taux d'élimination, pas de la quantité de départ — et c'est lui qu'il faut comparer au seuil de toxicité.

**Réponse.** a)  $d_{n+1} = 0,7 d_n + 40$  · b) 110 ; 117 ; 121,9 · c) environ 133,33 mg.

### Correction de l'exercice 15 - Un modèle de population avec prélèvement

a) On applique la croissance, puis on retire le prélèvement.

$$p_{n+1} = 1,12 p_n - 250 \quad \text{avec} \quad p_0 = 3\,000$$

- $p_1 = 1,12 \times 3\,000 - 250 = 3\,360 - 250 = 3\,110$ .
- $p_2 = 1,12 \times 3\,110 - 250 = 3\,483,2 - 250 = 3\,233,2$ .

b) Le nombre d'années est inconnu : boucle *while* avec compteur.

```
def annee():
    p = 3000
    n = 0
    while p <= 5000:
        p = 1.12 * p - 250
        n = n + 1
    return n
```

- Valeurs successives :  $p_8 \approx 4\,352,97$ ,  $p_9 \approx 4\,625,32$ ,  $p_{10} \approx 4\,930,36$ ,  $p_{11} \approx 5\,272$ .
- La fonction renvoie 11 : la population dépasse 5 000 au bout de 11 ans.

**Une vérification qui éclaire le modèle.** Le point d'équilibre  $L$  vérifie  $L = 1,12L - 250$ , donc  $0,12L = 250$  et  $L \approx 2\,083$ .

- La population de départ (3 000) est **au-dessus** de cet équilibre : elle s'en éloigne donc et croît indéfiniment.
- Si elle était partie de 2 000, elle aurait au contraire décru jusqu'à l'extinction. Le même prélèvement peut donc être soutenable ou fatal selon la population initiale.

**Réponse.** a)  $p_{n+1} = 1,12 p_n - 250$ ,  $p_1 = 3\,110$  et  $p_2 = 3\,233,2$  · b) 11 ans.

### Correction de l'exercice 16 - Synthèse : renouveler un parc de machines

a) Si 12 % des machines sont mises au rebut, il en reste  $100 - 12 = 88\%$ , soit un facteur 0,88 ; on ajoute ensuite les 7 machines neuves.

- $m_{n+1} = 0,88 m_n + 7$ .
- $m_1 = 0,88 \times 80 + 7 = 70,4 + 7 = 77,4$ .
- $m_2 = 0,88 \times 77,4 + 7 = 68,112 + 7 = 75,112$ .

b) Le nombre de tours est donné par  $n$  : boucle *for*.

```
def parc(n):
    m = 80
    for i in range(n):
        m = 0.88 * m + 7
    return m
```

c) La valeur d'équilibre  $L$  vérifie  $L = 0,88L + 7$ .

$$0,12L = 7 \quad \text{donc} \quad L = \frac{7}{0,12} \approx 58,33$$

- Le parc se stabilise autour de 58 **machines**, en décroissant depuis 80.
- Ce nombre ne dépend **pas** de l'effectif initial : il ne fait intervenir que le taux de rebut et le nombre d'achats. Un atelier parti de 40 machines monterait vers la même valeur.

d) On cherche le nombre d'achats annuels  $a$  tel que l'équilibre atteigne 70 machines.

- L'équilibre vérifie  $L = 0,88L + a$ , soit  $0,12L = a$ .
- Avec  $L = 70$  :  $a = 0,12 \times 70 = 8,4$ .
- Il faut donc acheter 9 **machines par an** (on ne peut pas en acheter 8,4), ce qui porte l'équilibre à  $\frac{9}{0,12} = 75$  machines.
- **Conclusion pour l'atelier** : avec 8 achats annuels l'équilibre tomberait à 66,7 machines, en dessous de l'objectif. Le passage de 7 à 9 achats par an fait donc remonter le parc d'équilibre de 58 à 75 machines — l'effet d'un achat supplémentaire est bien plus fort que ce que suggère le chiffre brut, car il se cumule année après année.

**Réponse.** a)  $m_1 = 77,4$  et  $m_2 = 75,112$  · c) environ 58,33 machines, indépendamment de l'effectif initial · d) 9 achats par an.