

Plaquette 2 – Lire et dérouler un programme

Algorithmique et Python – Terminale STI2D

Corrections détaillées

Boîte à outils

Lire un programme Python.

- **Affectation** : $a = a + 1$ ne dit pas « a égale $a + 1$ » mais « remplacer la valeur de a par l'ancienne augmentée de 1 ». On calcule toujours la droite avec les **anciennes** valeurs.
- **Boucle bornée** : `for i in range(n)`: exécute le bloc n fois, avec i prenant les valeurs $0, 1, \dots, n - 1$ — jamais n .
- `range(a, b)` donne $a, a + 1, \dots, b - 1$; `range(a, b, p)` avance de p en p et s'arrête **avant** b .
- **Boucle non bornée** : `while condition`: recommence tant que la condition est vraie. Elle est testée **avant** chaque tour ; si elle est fausse dès le départ, le bloc n'est jamais exécuté.
- **Indentation** : le décalage détermine ce qui est dans la boucle. Un `return` indenté dans une boucle sort de la fonction **dès le premier tour**.
- **Comparaisons** : $<$ est strict, $<=$ est large. La frontière est la source d'erreur la plus fréquente.
- **Listes** : `L[0]` est le premier élément, `len(L)` le nombre d'éléments, et `for x in L`: parcourt les valeurs une à une.
- **Accumulateur** : une variable s qui part de 0 et reçoit un terme à chaque tour calcule une somme :

$$s = u_0 + u_1 + \dots + u_{n-1} = \sum_{i=0}^{n-1} u_i$$

- **Méthode du tableau de variables** : une colonne par variable, une ligne par tour, plus une ligne pour l'état initial. On s'arrête quand la condition d'arrêt devient fausse — en notant bien la valeur qui a provoqué l'arrêt.

Correction de l'exercice 1 – Une somme dans une boucle for

`range(5)` fournit les entiers 0, 1, 2, 3, 4 : cinq tours, et jamais la valeur 5.

Tour	i	s après le tour
initial	—	0
1	0	0
2	1	1
3	2	3
4	3	6
5	4	10

- La fonction renvoie donc 10. **Contrôle** : la fonction calcule la somme des entiers de 0 à 4,

$$\sum_{i=0}^4 i = 0 + 1 + 2 + 3 + 4 = 10$$

— **Piège classique** : croire que $\text{range}(5)$ va jusqu'à 5 et répondre 15.

Réponse. 10.

Correction de l'exercice 2 - Compter les tours d'une boucle while

On déroule en notant la valeur de u **après** chaque tour.

Tour	u	n
initial	3	0
1	7	1
2	15	2
3	31	3
4	63	4
5	127	5

- Après le cinquième tour, $u = 127$: la condition $u < 100$ devient fausse, la boucle s'arrête.
- La fonction renvoie $n = 5$.
- **Attention** : le dernier tour est bien compté. C'est lui qui fait franchir le seuil, et beaucoup d'élèves l'oublient en répondant 4.

Réponse. 5.

Correction de l'exercice 3 - Deux variables qui évoluent ensemble

La difficulté est l'ordre des deux affectations : a est modifié **avec l'ancienne valeur** de b , puis b change.

Tour	a	b
initial	1	6
1	7	4
2	11	2
3	13	0

- Détail du tour 1 : $a = 1 + 6 = 7$, puis $b = 6 - 2 = 4$.
- Détail du tour 2 : $a = 7 + 4 = 11$, puis $b = 4 - 2 = 2$.
- Détail du tour 3 : $a = 11 + 2 = 13$, puis $b = 2 - 2 = 0$. À la fin, a a reçu les trois valeurs successives de b , et b a perdu trois fois 2 :

$$a = 1 + 6 + 4 + 2 = 13 \quad b = 6 - 3 \times 2 = 0$$

- **Piège** : si l'on inversait les deux lignes, on trouverait $a = 1 + 4 + 2 + 0 = 7$. L'ordre des affectations change le résultat.

Réponse. $a = 13$ et $b = 0$.

Correction de l'exercice 4 - Frontières d'un test

Les tests sont examinés **dans l'ordre**, et le premier vrai décide.

- $x = 9,5$: $9,5 < 10$ est vrai, la fonction renvoie *faible*.

- $x = 10$: $10 < 10$ est **faux** (l'inégalité est stricte), on passe au test suivant ; $10 < 20$ est vrai, donc *moyen*.
- $x = 20$: $20 < 10$ faux, $20 < 20$ faux, on tombe dans le *else*, donc *fort*.
- **À retenir** : une valeur exactement égale à une borne ne vérifie jamais une inégalité stricte. Pour ranger 10 dans « faible », il faudrait écrire *if* $x \leq 10$.

Réponse. *faible, moyen, fort.*

Correction de l'exercice 5 - Un return qui interrompt la boucle

Le *return* placé **dans** la boucle interrompt la fonction dès qu'il est atteint.

Cas $n = 10$. On teste $i^2 > 30$ pour $i = 0, 1, 2, \dots$

- $i = 5$: $25 > 30$ est faux, on continue.
- $i = 6$: $36 > 30$ est vrai, la fonction renvoie immédiatement 6 — les valeurs 7, 8, 9 ne sont jamais examinées.

Cas $n = 5$. La boucle ne va que jusqu'à $i = 4$, et $16 > 30$ est faux.

- Aucun *return* n'est déclenché dans la boucle : on atteint la dernière ligne et la fonction renvoie -1 .
- **Rôle du** -1 : c'est une valeur sentinelle qui signale « aucune solution trouvée ». Sans elle, la fonction ne renverrait rien dans ce cas.

Réponse. $h(10)$ renvoie 6 et $h(5)$ renvoie -1 .

Correction de l'exercice 6 - Une erreur d'indentation

Diagnostic. Le *return* est indenté au même niveau que $s = s + i$: il se trouve donc **dans** la boucle.

- Au premier tour, $i = 1$ donne $s = 1$, puis le *return* est exécuté : la fonction s'arrête et renvoie 1.
- Les tours suivants n'ont jamais lieu.

Correction. Il faut sortir le *return* de la boucle en le désindentant d'un niveau.

```
def somme(n):
    s = 0
    for i in range(1, n + 1):
        s = s + i
    return s
```

- Vérification : *somme(5)* parcourt $i = 1, 2, 3, 4, 5$ et renvoie $1 + 2 + 3 + 4 + 5 = 15$.
- **À retenir** : en Python, l'indentation n'est pas de la présentation, c'est de la syntaxe. Un décalage de quatre espaces change le sens du programme.

Réponse. Le *return* est dans la boucle ; il faut le désindenter pour obtenir 15.

Correction de l'exercice 7 - Repérer une boucle infinie

Diagnostic. La condition d'arrêt demande que u descende sous 10, mais l'instruction $u = u * 1.2$ **augmente** u de 20 % à chaque tour.

- u vaut successivement 60, 72, 86,4, ... : la condition $u > 10$ reste vraie pour toujours.
- Une boucle *while* ne s'arrête que si le corps de la boucle fait **progresser vers** la condition d'arrêt.

Correction. Pour une baisse, le coefficient doit être inférieur à 1, par exemple 0,8 pour une baisse de 20 %.

```
def seuil():
    u = 50
    n = 0
    while u > 10:
        u = u * 0.8
        n = n + 1
    return n
```

- Contrôle : $50 \times 0,8^7 \approx 10,49$ (encore au-dessus) et $50 \times 0,8^8 \approx 8,39$ (en dessous).
- La fonction corrigée renvoie donc 8.
- **Réflexe à prendre** : devant une boucle *while*, toujours se demander « qu'est-ce qui rapproche de l'arrêt ? ». Si la réponse est « rien », la boucle est infinie.

Réponse. Le facteur 1,2 fait croître u ; avec 0,8 la fonction renvoie 8.

Correction de l'exercice 8 - Une somme de termes de suite

La variable s accumule les termes **avant** que u ne soit modifié : on additionne donc u_0 à u_5 .

i	u ajouté	s cumulé
0	800	800
1	600	1 400
2	450	1 850
3	337,5	2 187,5
4	253,125	2 440,625
5	189,843 75	2 630,468 75

- La fonction renvoie 2 630,468 75.
- **Contrôle par la formule du cours.** La somme des 6 premiers termes d'une suite géométrique de premier terme 800 et de raison 0,75 vaut :

$$S = 800 \times \frac{1 - 0,75^6}{1 - 0,75}$$

- $0,75^6 \approx 0,177\,979$, donc $S \approx 800 \times \frac{0,822\,021}{0,25} = 800 \times 3,288\,086 \approx 2\,630,47$. Cohérent.
- **Piège** : si les deux lignes étaient inversées, la somme porterait sur u_1 à u_6 et vaudrait 1 972,85.

Réponse. 2 630,468 75.

Correction de l'exercice 9 - Un compteur conditionnel

La boucle *for* x in L : parcourt les **valeurs** de la liste, une par une, et incrémente c quand la valeur dépasse strictement 20.

Valeur	12	25	20	31	8	47
$x > 20$?	non	oui	non	oui	non	oui

- Attention à 20 : l'inégalité est **stricte**, donc 20 n'est pas compté.
- Trois valeurs conviennent : 25, 31 et 47.
- La fonction renvoie 3.
- Remarque : avec *if* $x \geq 20$:, la réponse serait 4. Lire l'énoncé (« au moins », « plus de ») est donc aussi important que lire le code.

Réponse. 3.

Correction de l'exercice 10 - Parcourir une liste par ses indices

$\text{len}(L)$ vaut 4, donc i prend les valeurs 0, 1, 2, 3 : ce sont les **indices** de la liste, et non ses valeurs.

i	0	1	2	3
$L[i]$	4	9	2	7
$L[i] \times i$	0	9	4	21

- $s = 0 + 9 + 4 + 21 = 34$.
- **À retenir** : en Python, le premier élément porte l'indice 0 et le dernier l'indice $\text{len}(L) - 1$. Ici $L[0]$ vaut 4 et $L[3]$ vaut 7 ; $L[4]$ provoquerait une erreur.
- Deux parcours possibles : *for* x *in* L : donne les valeurs, *for* i *in* $\text{range}(\text{len}(L))$: donne les indices. On choisit le second quand la position compte, comme ici.

Réponse. $s = 34$.

Correction de l'exercice 11 - Le maximum d'une liste

a) On suit la variable m , qui retient le plus grand élément rencontré.

- Départ : $m = 3$.
- $x = 3$: $3 > 3$ faux, m reste 3.
- $x = 17$: vrai, $m = 17$.
- $x = 8$ puis $x = 12$: faux les deux fois.
- La fonction renvoie 17.

b) Avec une initialisation à 0, la fonction échouerait sur une liste de nombres **tous négatifs**.

- Exemple : sur $[-5, -2, -9]$, aucun élément ne dépasse 0, et la fonction renverrait 0 — qui n'appartient même pas à la liste.
- En partant de $L[0]$, le résultat est toujours un élément de la liste. C'est la bonne pratique.
- **Limite** : la fonction plante sur une liste vide, puisque $L[0]$ n'existe pas.

Réponse. a) 17 · b) sinon la fonction se trompe sur une liste de nombres négatifs.

Correction de l'exercice 12 - Calculer une moyenne

La fonction additionne d'abord toutes les valeurs, puis divise par leur nombre.

- Somme : $12,5 + 14 = 26,5$; $+11,5 = 38$; $+15 = 53$; $+12 = 65$.
- Nombre de valeurs : $\text{len}(L)$ vaut 5.
- La moyenne est la somme divisée par l'effectif :

$$\bar{x} = \frac{65}{5} = 13$$

- La fonction renvoie 13.
- ***Position du return*** : il est hors de la boucle, ce qui est indispensable. Indenté dans la boucle, il renverrait $\frac{12,5}{5} = 2,5$ dès le premier tour.

Réponse. 13.

Correction de l'exercice 13 - Deux boucles imbriquées

Premier programme. La boucle interne fait 3 tours, et elle est relancée entièrement à chacun des 4 tours de la boucle externe.

— $c = 4 \times 3 = 12$.

Second programme. Le nombre de tours de la boucle interne dépend de i .

i	0	1	2	3
tours de la boucle interne	0	1	2	3

— Pour $i = 0$, $\text{range}(0)$ est vide : la boucle interne n'est pas exécutée du tout.

— On additionne les tours de la boucle interne :

$$c = 0 + 1 + 2 + 3 = 6$$

— **À retenir** : avec des boucles imbriquées, on compte les tours **internes**, pas les lignes de code. Quand la borne interne dépend de l'indice externe, le total n'est plus un produit mais une somme.

Réponse. 12 pour le premier, 6 pour le second.

Correction de l'exercice 14 - Un range avec un pas

$\text{range}(a, b, p)$ part de a , avance de p en p , et s'arrête **avant** d'atteindre b .

Premier programme : $a = 2$, $b = 11$, $p = 3$.

— Valeurs : 2, 5, 8. La valeur suivante, 11, est exclue car la borne n'est jamais atteinte.

— $s = 2 + 5 + 8 = 15$.

Second programme : le pas est **négatif**, la boucle descend.

— Valeurs : 10, 8, 6, 4, 2. La valeur 0 est exclue pour la même raison.

— On additionne les valeurs prises par i :

$$s = 10 + 8 + 6 + 4 + 2 = 30$$

— **Piège** : avec un pas négatif, on descend jusqu'à $b + 1$ compris. Ici la boucle s'arrête à 2 et non à 0 : c'est la source d'erreur la plus fréquente sur range .

Réponse. 2 ; 5 ; 8 et $s = 15$ — puis 10 ; 8 ; 6 ; 4 ; 2 et $s = 30$.

Correction de l'exercice 15 - Un tarif par paliers

a) On regarde d'abord dans quel palier tombe n .

— $n = 80 \leq 100$: premier palier, $2,5 \times 80 = 200$ €.

— $n = 100 \leq 100$: l'inégalité est **large**, donc encore le premier palier, $2,5 \times 100 = 250$ €.

— $n = 150 > 100$: second palier, $250 + 2 \times (150 - 100) = 250 + 100 = 350$ €.

b) Il faut comparer les deux formules exactement au point de rupture, $n = 100$.

— Premier palier en $n = 100$: $2,5 \times 100 = 250$ €.

— Second palier en $n = 100$: $250 + 2 \times 0 = 250$ €.

— Les deux valeurs coïncident : le tarif est **continu**, il n'y a pas de saut.

— **Lecture commerciale** : au-delà de 100 pièces, chaque pièce supplémentaire ne coûte plus que 2 € au lieu de 2,50 €. C'est un tarif dégressif, construit pour qu'un client ne paie jamais moins en commandant plus.

Réponse. a) 200 € ; 250 € ; 350 € · b) les deux formules donnent 250 € en $n = 100$.

Correction de l'exercice 16 - Synthèse : un bilan de production

a) - n compte les pièces dont l'épaisseur dépasse **strictement** le seuil : c'est le nombre de pièces hors tolérance.

— $s / \text{len}(L)$ est la somme des épaisseurs divisée par leur nombre : c'est l'épaisseur **moyenne** du lot.

b) On parcourt la liste.

— Somme : $48 + 52 + 61 + 47 + 55 + 63 + 50 + 44 = 420$.

— Nombre de valeurs : 8, donc moyenne = $\frac{420}{8} = 52,5 \mu\text{m}$.

— Valeurs strictement supérieures à 55 : 61 et 63. Attention, 55 n'est **pas** compté.

— La fonction renvoie donc $n = 2$ et une moyenne de 52,5.

c) Il suffit d'ajouter une variable qui retient le maximum, initialisée avec le premier élément.

```
def bilan(L, seuil):
    n = 0
    s = 0
    m = L[0]
    for x in L:
        s = s + x
        if x > seuil:
            n = n + 1
        if x > m:
            m = x
    return n, s / len(L), m
```

— Sur le lot donné, $m = 63 \mu\text{m}$.

d) Le nombre de pièces hors tolérance est n ; le **pourcentage** s'obtient en le rapportant à l'effectif.

— Remplacer `return n, s / len(L)` par `return 100 * n / len(L), s / len(L)`.

— Sur ce lot : $\frac{100 \times 2}{8} = 25$, soit 25 % de pièces hors tolérance.

— **Lecture industrielle** : la moyenne ($52,5 \mu\text{m}$) est confortablement sous le seuil, et pourtant un quart du lot le dépasse. Une moyenne correcte ne garantit jamais la conformité : c'est la **dispersion** qu'il faut surveiller.

Réponse. a) le nombre de pièces hors tolérance et l'épaisseur moyenne · b) $n = 2$ et $52,5 \mu\text{m}$ · c) ajouter une variable m initialisée à $L[0]$ · d) renvoyer $100 * n / \text{len}(L)$, soit 25 %.