Algorithmique et Python — Première STI2D

20 exercices corrigés · mathlvl.fr

Énoncés et corrigés détaillés

Présentation

En STI2D, Python n'est pas une matière à part : c'est l'outil qui permet de faire ce qu'on ne sait pas faire à la main — chercher un seuil sur une suite, balayer un intervalle pour résoudre une équation, simuler dix mille prélèvements pour voir fluctuer une fréquence. Trois mécanismes suffisent à presque tout : le COMPTEUR (`c = c + 1`), l'ACCUMULATEUR (`S = S + ...`) et la boucle `while` pour un arrêt conditionnel. Ces 20 exercices corrigés vont de la trace d'affectations à la simulation de Monte-Carlo, en passant par le débogage d'un programme et la lecture d'une boucle. Chaque correction détaille l'évolution des variables, tour par tour. La difficulté est croissante : l'énoncé puis la correction détaillée s'affichent d'un clic.

Version en ligne, avec les corrections dépliables :
mathlvl.fr/exercices-corriges/premiere-sti2d/algorithmique-python-sti2d

Exercice 1 – Suivre des affectations

On exécute les instructions suivantes :

```
a = 5
b = 12
a = a + b
b = a - b
```

1. Donne la valeur de `a` et de `b` après exécution.

Exercice 2 – Que renvoie cette fonction ?

On définit :

```
def f(x):
    return 3*x - 7
```

1. Que renvoie `f(4)` ? `f(0)` ? `f(-2)` ?

2. Quelle valeur de `x` donne `f(x)` égal à 20 ?

Exercice 3 – Compléter une fonction de conversion

Une pression donnée en bars doit être convertie en pascals, sachant que 1 bar = 100 000 Pa. On veut compléter :

```
def enPascals(p):
    return ...
```

1. Complète la ligne manquante.

2. Écris la fonction inverse, qui convertit des pascals en bars.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Exercice 4 – Boucle bornée : une somme

On exécute :

```
S = 0
for k in range(1, 6):
    S = S + k
```

1. Donne la trace de S tour par tour.

2. Que vaut S à la fin ?

Exercice 5 – Instruction conditionnelle

Une pièce est conforme si sa longueur ℓ vérifie $119{,}6 \leq \ell \leq 120{,}4$ mm. On propose :

```
def controle(L):
    if L >= 119.6 and L <= 120.4:
        return "conforme"
    else:
        return "rebut"
```

1. Que renvoie `controle(120.1)` ? `controle(119.4)` ? `controle(120.4)` ?

2. Réécris la condition avec un double encadrement.

Exercice 6 – Compter avec un compteur

Une liste contient les longueurs mesurées de pièces :

```
L = [120.1, 119.4, 120.3, 120.7, 119.9, 120.4]
```

On veut compter les pièces hors tolérance (hors de $[119{,}6\,;\,120{,}4]$).

1. Écris une fonction qui renvoie ce nombre.

2. Applique-la à la liste donnée en détaillant.

Exercice 7 – Moyenne d'une liste de mesures

On veut calculer la moyenne des valeurs d'une liste, sans utiliser de fonction toute faite.

1. Écris la fonction.

2. Applique-la à `[22, 41, 58, 79, 100]`.

Exercice 8 – Chercher le maximum

On veut trouver la plus grande valeur d'une liste non vide, sans utiliser `max`.

1. Écris la fonction.

2. Applique-la à `[14, 9, 23, 23, 7, 19]` en détaillant la trace.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Exercice 9 – Boucle non bornée : atteindre un seuil

Une machine perd 8% de sa valeur par an. Elle vaut 30 000 € à l'achat. On veut savoir au bout de combien d'années elle vaut moins de 15 000 €.

1. Écris le programme.

2. Donne la trace des premières itérations, puis le résultat.

Exercice 10 – Lire un programme et prévoir sa sortie

On exécute :

```
u = 2
for i in range(4):
    u = 3 * u - 1
```

1. Donne la trace de `u`.

2. Quelle suite ce programme calcule-t-il ?

Exercice 11 – Tableau de valeurs d'une fonction

On veut afficher un tableau de valeurs de $f(x) = x^2 - 6x + 5$ sur $[0\,;6]$ avec un pas de 0,5.

1. Écris le programme.

2. Quelles valeurs de f obtient-on pour $x = 1$, $x = 3$ et $x = 5$?

Exercice 12 – Résoudre une équation par balayage

On cherche une valeur approchée de la solution de $x^2 = 5$ dans $[2\,;3]$, à 0,01 près.

1. Écris un programme de balayage.

2. Donne la valeur obtenue et vérifie-la.

Exercice 13 – Simuler un lancer de dé

On rappelle que `randint(1, 6)` du module `random` renvoie un entier au hasard entre 1 et 6 inclus. On veut estimer la probabilité d'obtenir un 6.

1. Écris une fonction qui simule N lancers et renvoie la fréquence de 6.

2. Quelle valeur doit-elle renvoyer pour N grand ?

Exercice 14 – Construire une liste de termes

On veut construire la liste des 8 premiers termes de la suite définie par $u_0 = 500$ et $u_{n+1} = 0{,}9\,u_n + 30$.

1. Écris le programme utilisant `append`.

2. Donne les quatre premiers termes.

3. Vers quelle valeur la suite semble-t-elle se stabiliser ?

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Exercice 15 – Chercher un minimum par balayage

On veut localiser le minimum de $f(x) = x^2 - 6x + 11$ sur $[0\,;6]$ en balayant l'intervalle avec un pas de 0,1, sans utiliser la dérivée.

1. Écris un programme renvoyant l'abscisse du minimum trouvé et la valeur correspondante.

2. Quel résultat renvoie-t-il ?

3. Vérifie par le calcul.

Exercice 16 – Type bac : simulation de Monte-Carlo

Une production contient 12% de pièces non conformes. On prélève des échantillons de 50 pièces et on veut estimer par simulation la probabilité qu'un échantillon contienne au moins 10 pièces non conformes. On rappelle que `random()` renvoie un réel de $[0\,;1[$.

1. Écris une fonction simulant un échantillon et renvoyant le nombre de pièces non conformes.

2. Écris une fonction estimant la probabilité cherchée sur N échantillons.

3. Combien de pièces non conformes attend-on en moyenne par échantillon ?

Exercice 17 – Type bac : seuil de rentabilité par programme

Le coût de production de q pièces est $C(q) = 0{,}5q^2 + 20q + 800$ euros, et chaque pièce est vendue 60 €.

1. Écris une fonction donnant le bénéfice pour q pièces.

2. Écris un programme déterminant la plus petite quantité rendant le bénéfice positif.

3. Donne cette quantité en détaillant.

Exercice 18 – Type bac : déboguer un programme

Un élève veut compter les valeurs supérieures à 100 dans une liste. Il écrit :

```
def compte(L):
  for x in L:
    c = 0
  if x > 100:
    c = c + 1
  return c
```

1. Que renvoie ce programme pour `[150, 80, 200]` ?

2. Identifie les deux erreurs et corrige le programme.

Exercice 19 – Type bac : programme de statistique à deux variables

Deux listes de même longueur contiennent les tensions X (en V) et les pressions Y (en kPa) relevées lors d'un étalonnage :

```
X = [1, 2, 3, 4, 5]
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
Y = [22, 41, 58, 79, 100]
```

1. Écris une fonction renvoyant les coordonnées du point moyen.

2. Applique-la à ces listes.

3. Écris une fonction qui, pour une droite d'équation $y = ax + b$, renvoie la somme des carrés des résidus.

Exercice 20 – Type bac : observer la fluctuation d'échantillonnage

Une production contient une proportion $p = 0{,}2$ de pièces de premier choix. On veut observer comment la fréquence observée fluctue selon la taille n de l'échantillon.

1. Écris une fonction renvoyant la fréquence de pièces de premier choix dans un échantillon de taille n.

2. Écris une fonction renvoyant l'écart maximal à 0,2 observé sur 100 échantillons de taille n.

3. Qu'observe-t-on en passant de $n = 20$ à $n = 500$?

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Corrigés détaillés

Les corrections des 20 exercices, dans l'ordre des énoncés.

Correction de l'exercice 1 – Suivre des affectations

Une affectation `x = ...` calcule d'abord le membre de DROITE, puis range le résultat dans la variable de gauche. L'ancienne valeur est perdue.

Après `a = 5` : `a` vaut 5.

Après `b = 12` : `a` = 5 et `b` = 12.

`a = a + b` : on calcule 5 + 12 = 17, puis on range 17 dans `a`. Donc `a` = 17 et `b` = 12.

`b = a - b` : on calcule 17 − 12 = 5 avec les valeurs ACTUELLES, puis on range 5 dans `b`. Donc `a` = 17 et `b` = 5.

Remarque : `b` a pris l'ancienne valeur de `a`. Ces deux lignes sont le début de l'astuce classique d'échange de deux variables sans variable auxiliaire.

Réponse. `a` = 17 et `b` = 5.

Rappel de cours. `x = expression` n'est pas une égalité mathématique : c'est une AFFECTATION, qui écrase l'ancienne valeur.

Erreur fréquente. Utiliser l'ancienne valeur de `a` (5) dans la dernière ligne : à ce moment `a` vaut déjà 17.

Correction de l'exercice 2 – Que renvoie cette fonction ?

La fonction calcule $3x - 7$ et renvoie le résultat.

`f(4)` : $3 \times 4 - 7 = 12 - 7 = 5$.

`f(0)` : $3 \times 0 - 7 = -7$.

`f(-2)` : $3 \times (-2) - 7 = -6 - 7 = -13$.

Pour la dernière question, on résout l'équation $3x - 7 = 20$.

$3x = 27$, donc $x = 9$.

Contrôle : `f(9)` renvoie $3 \times 9 - 7 = 20$ ✔.

Réponse. `f(4)` = 5 · `f(0)` = −7 · `f(-2)` = −13 · `f(9)` = 20.

Rappel de cours. Une fonction Python est l'exact équivalent d'une fonction mathématique : `f(x)` c'est $f(x)$.

Erreur fréquente. Oublier les parenthèses mentales sur les négatifs : $3 \times (-2)$ vaut −6, et non $-3 \times 2 - 7 = -13$ par hasard juste ici.

Correction de l'exercice 3 – Compléter une fonction de conversion

Convertir des bars en pascals, c'est multiplier par 100 000.

```
def enPascals(p):
    return 100000 * p
```

Contrôle : `enPascals(2.5)` renvoie 250 000 Pa, ce qui est correct pour 2,5 bars ✔.

La conversion inverse divise par 100 000 :

```
def enBars(q):
    return q / 100000
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Contrôle croisé : `enBars(enPascals(2.5))` renvoie 2,5 ✔. Les deux fonctions se défont l'une l'autre.

Détail d'écriture : en Python, le séparateur décimal est le POINT (`2.5`), et jamais la virgule — `f(2,5)` serait interprété comme deux arguments.

Réponse. `return 100000 * p` · fonction inverse : `return q / 100000`.

Rappel de cours. Sans `return`, une fonction ne renvoie rien (`None`) : l'affichage par `print` n'est pas un retour de valeur.

Erreur fréquente. Écrire la virgule décimale : `2,5` n'est pas un nombre en Python. On écrit `2.5`.

Correction de l'exercice 4 – Boucle bornée : une somme

`range(1, 6)` parcourt 1, 2, 3, 4, 5 : la borne de droite est EXCLUE.

Avant la boucle : `S` = 0.

$k = 1$: `S` = 0 + 1 = 1.

$k = 2$: `S` = 1 + 2 = 3.

$k = 3$: `S` = 3 + 3 = 6.

$k = 4$: `S` = 6 + 4 = 10.

$k = 5$: `S` = 10 + 5 = 15.

À la fin, `S` = 15, c'est-à-dire 1 + 2 + 3 + 4 + 5.

`S` est un ACCUMULATEUR : une variable initialisée à 0 qui reçoit une contribution à chaque tour.

Réponse. Trace : 1, 3, 6, 10, 15 · `S` = 15 à la fin.

Rappel de cours. `range(a, b)` parcourt a, $a + 1$, …, $b - 1$: la borne supérieure est exclue.

Erreur fréquente. Croire que `range(1, 6)` inclut 6 : on annoncerait 21 au lieu de 15.

Correction de l'exercice 5 – Instruction conditionnelle

`controle(120.1)` : 120,1 ≥ 119,6 est vrai ET 120,1 ≤ 120,4 est vrai, donc la fonction renvoie « conforme ».

`controle(119.4)` : 119,4 ≥ 119,6 est FAUX, la condition globale est fausse, donc « rebut ».

`controle(120.4)` : 120,4 ≤ 120,4 est vrai (inégalité LARGE), donc « conforme ». La borne est incluse, conformément à la tolérance.

Python accepte l'écriture directe d'un encadrement :

```
if 119.6 <= L <= 120.4:
```

Cette forme est plus lisible et strictement équivalente à la conjonction avec `and`.

Attention à ne pas utiliser `or` : avec `if L >= 119.6 or L <= 120.4`, la condition serait vraie pour TOUTE valeur, et toutes les pièces seraient déclarées conformes.

Réponse. « conforme », « rebut », « conforme » · condition équivalente :
`if 119.6 <= L <= 120.4:`

Rappel de cours. `and` exige les deux conditions · `or` en exige au moins une. Un encadrement se traduit toujours par `and`.

Erreur fréquente. Confondre = (affectation) et == (test d'égalité) dans une condition :

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Python renvoie une erreur de syntaxe.

Correction de l'exercice 6 – Compter avec un compteur

On parcourt la liste et on incrémente un compteur à chaque valeur hors tolérance.

```
def horsTolerance(L):
    c = 0
    for x in L:
        if x < 119.6 or x > 120.4:
            c = c + 1
    return c
```

Ici la condition utilise `or` : être hors tolérance, c'est être trop petit OU trop grand.

Examen des valeurs : 120,1 est dans la tolérance ; 119,4 < 119,6 → hors tolérance (`c` = 1).

120,3 est dans la tolérance ; 120,7 > 120,4 → hors tolérance (`c` = 2).

119,9 est dans la tolérance ; 120,4 est exactement à la borne, donc DANS la tolérance.

La fonction renvoie `c` = 2.

Remarque : `for x in L` parcourt directement les VALEURS de la liste. On aurait pu écrire `for i in range(len(L))` et utiliser `L[i]`, mais c'est plus lourd.

Réponse. Compteur incrémenté si `x < 119.6 or x > 120.4` · la fonction renvoie 2 (les valeurs 119,4 et 120,7).

Rappel de cours. Un COMPTEUR s'initialise à 0 avant la boucle et s'incrémente de 1 dans le `if`.

Erreur fréquente. Initialiser `c = 0` À L'INTÉRIEUR de la boucle : le compteur repartirait de zéro à chaque tour et ne pourrait jamais dépasser 1.

Correction de l'exercice 7 – Moyenne d'une liste de mesures

On additionne toutes les valeurs avec un accumulateur, puis on divise par le nombre de valeurs.

```
def moyenne(L):
    S = 0
    for x in L:
        S = S + x
    return S / len(L)
```

`len(L)` donne le nombre d'éléments de la liste, ici 5.

Trace de l'accumulateur : 0 → 22 → 63 → 121 → 200 → 300.

Division finale : $\frac{300}{5} = 60$.

La fonction renvoie 60, ce qui est bien la moyenne de ces cinq mesures ✔.

Point important : la division est APRÈS la boucle. Placée à l'intérieur, elle diviserait des sommes partielles et le résultat n'aurait aucun sens.

Réponse. Accumulateur puis `return S / len(L)` · la fonction renvoie 60.

Rappel de cours. Structure d'un calcul de moyenne : accumuler DANS la boucle, diviser APRÈS la boucle.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Erreur fréquente. Diviser par `len(L)` à chaque tour, ou oublier la division et renvoyer la somme 300.

Correction de l'exercice 8 – Chercher le maximum

Principe : on retient une valeur « championne », initialisée au PREMIER élément, et on la remplace dès qu'on trouve mieux.

```
def maximum(L):
    m = L[0]
    for x in L:
        if x > m:
            m = x
    return m
```

Initialisation : `m` = `L[0]` = 14.

$x = 14$: pas plus grand, `m` reste 14. $x = 9$: non, `m` = 14.

$x = 23$: plus grand, donc `m` = 23.

$x = 23$ à nouveau : 23 > 23 est FAUX, `m` reste 23 (ce qui ne change rien).

$x = 7$ puis $x = 19$: pas plus grands. La fonction renvoie 23 ✔.

Pourquoi initialiser à `L[0]` et non à 0 ? Parce qu'avec une liste de valeurs négatives (par exemple des écarts de réglage), l'initialisation à 0 renverrait 0, qui n'appartient même pas à la liste.

Réponse. Champion initialisé à `L[0]`, remplacé si `x > m` · la fonction renvoie 23.

Rappel de cours. Recherche de maximum : initialiser au premier élément de la liste, jamais à 0.

Erreur fréquente. Écrire `if x > m: return x` : la fonction s'arrêterait au premier élément supérieur à `L[0]`, sans finir le parcours.

Correction de l'exercice 9 – Boucle non bornée : atteindre un seuil

Le nombre d'itérations est inconnu à l'avance : c'est un cas de boucle `while`.

```
def duree():
    v = 30000
    n = 0
while v >= 15000:
    v = 0.92 * v
    n = n + 1
    return n
```

Trace : $n = 1 \to v = 27\,600$; $n = 2 \to v = 25\,392$; $n = 3 \to v \approx 23\,360{,}6$; $n = 4 \to v \approx 21\,491{,}8$.

Puis $n = 5 \to v \approx 19\,772{,}5$; $n = 6 \to v \approx 18\,190{,}7$; $n = 7 \to v \approx 16\,735{,}5$; $n = 8 \to v \approx 15\,396{,}6$.

15 396,6 est encore ≥ 15 000 : la boucle continue. $n = 9 \to v \approx 14\,164{,}9$, cette fois inférieur à 15 000 : la boucle s'arrête.

La fonction renvoie 9 : il faut 9 ans pour que la machine vaille moins de 15 000 €.

Contrôle par le calcul direct : $30\,000 \times 0{,}92^{8} \approx 15\,396$ et $30\,000 \times 0{,}92^{9} \approx 14\,165$ ✔.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Réponse. Boucle `while v >= 15000` avec `v = 0.92*v` · la fonction renvoie 9 années.

Rappel de cours. `while` : nombre d'itérations inconnu, arrêt sur CONDITION · `for` : nombre d'itérations fixé d'avance.

Erreur fréquente. Incrémenter `n` avant de modifier `v` n'a pas d'importance ici, mais oublier `n = n + 1` provoque une boucle infinie… et un programme qui ne rend jamais la main.

Correction de l'exercice 10 – Lire un programme et prévoir sa sortie

`range(4)` parcourt 0, 1, 2, 3 : la boucle tourne 4 fois.

Départ : u = 2.

Tour 1 : u = 3 × 2 − 1 = 5.

Tour 2 : u = 3 × 5 − 1 = 14.

Tour 3 : u = 3 × 14 − 1 = 41.

Tour 4 : u = 3 × 41 − 1 = 122.

À la fin, u = 122.

Ce programme calcule les termes de la suite définie par $u_0 = 2$ et $u_{n+1} = 3u_n - 1$: la variable u contient u_4 à la sortie.

Noter que la variable de boucle `i` n'apparaît pas dans le calcul : elle ne sert qu'à COMPTER les répétitions.

Réponse. Trace 2 → 5 → 14 → 41 → 122 · le programme calcule u_4 pour $u_0 = 2$ et $u_{n+1} = 3u_n - 1$.

Rappel de cours. `range(n)` parcourt 0 à $n - 1$, donc n valeurs : la boucle tourne n fois.

Erreur fréquente. Compter 5 tours pour `range(4)` : on obtiendrait 365 au lieu de 122.

Correction de l'exercice 11 – Tableau de valeurs d'une fonction

Le pas n'étant pas entier, on ne peut pas le mettre directement dans `range` (qui n'accepte que des entiers). On boucle sur un compteur entier et on reconstruit x.

```
def f(x):
    return x**2 - 6*x + 5
for k in range(13):
    x = 0.5 * k
    print(x, f(x))
```

Pourquoi `range(13)` ? De 0 à 6 avec un pas de 0,5, il y a $\frac{6}{0{,}5} + 1 = 13$ valeurs (les deux bornes comprises).

Notation : `x**2` désigne x^2 en Python (l'accent circonflexe `\^{}` ne désigne PAS la puissance).

$f(1) = 1 - 6 + 5 = 0$.

$f(3) = 9 - 18 + 5 = -4$: c'est le minimum de la fonction, au sommet de la parabole.

$f(5) = 25 - 30 + 5 = 0$.

Lecture du tableau : f s'annule en 1 et en 5 (les racines), et atteint -4 en 3, au milieu des racines ✔.

Réponse. Boucle `for k in range(13)` avec `x = 0.5*k` · $f(1) = 0$, $f(3) = -4$, $f(5) = 0$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Rappel de cours. `range` n'accepte que des entiers : pour un pas décimal, on boucle sur k et on pose $x = \text{pas} \times k$.

Erreur fréquente. Écrire `range(0, 6, 0.5)` : Python renvoie une erreur, le pas doit être entier.

Correction de l'exercice 12 – Résoudre une équation par balayage

Principe du balayage : on part de $x = 2$ et on avance par pas de 0,01 tant que x^2 reste inférieur à 5. Dès que x^2 dépasse 5, on s'arrête.

```
def balayage():
    x = 2
    while x**2 < 5:
        x = x + 0.01
    return x
```

La fonction est croissante sur $[2\,;3]$, ce qui garantit qu'on ne « rate » pas la solution : dès que le test échoue, on l'a dépassée.

Quelques valeurs : $2{,}23^2 = 4{,}972\,9 < 5$ (on continue) ; $2{,}24^2 = 5{,}017\,6 \geq 5$ (on s'arrête).

La fonction renvoie donc $x \approx 2{,}24$.

Encadrement : $2{,}23 < \sqrt{5} < 2{,}24$. La valeur exacte est $\sqrt{5} \approx 2{,}236\,07$ ✔, bien dans cet encadrement.

Précision : le programme renvoie la première valeur DÉPASSANT la solution. Pour un encadrement, on renverrait `x - 0.01` et `x`.

Remarque technique : les additions répétées de 0,01 accumulent de minuscules erreurs d'arrondi binaire. Pour 100 tours c'est indifférent, mais sur des millions d'itérations on préférerait `x = 2 + 0.01*k`.

Réponse. Balayage `while x**2 < 5` par pas de 0,01 · la fonction renvoie 2,24, et $2{,}23 < \sqrt{5} < 2{,}24$.

Rappel de cours. Le balayage exige une fonction MONOTONE sur l'intervalle, sinon on peut sauter par-dessus une solution.

Erreur fréquente. Utiliser `while x**2 <= 5` puis conclure à l'égalité : le programme ne trouve jamais la valeur exacte, seulement un encadrement.

Correction de l'exercice 13 – Simuler un lancer de dé

On compte les 6 obtenus, puis on divise par le nombre de lancers.

```
from random import randint
def frequence(N):
    c = 0
    for k in range(N):
        if randint(1, 6) == 6:
            c = c + 1
    return c / N
```

Attention à l'écriture du test d'égalité : == et non =.

La probabilité théorique d'obtenir 6 avec un dé équilibré est $\frac{1}{6} \approx 0{,}1667$.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Pour N grand, la fréquence renvoyée doit être proche de 0,1667. Avec $N = 100$, on peut obtenir 0,12 ou 0,21 ; avec $N = 1\,000\,000$, on sera typiquement à 0,001 près.

C'est la loi des grands nombres à l'œuvre : la fréquence observée se rapproche de la probabilité quand le nombre d'essais augmente.

Remarque : deux exécutions du programme donnent des résultats DIFFÉRENTS, puisque les tirages sont aléatoires. Un programme de simulation ne renvoie jamais deux fois la même valeur — c'est normal, ce n'est pas un bug.

Réponse. Compteur sur `randint(1, 6) == 6`, puis `return c / N` · pour N grand, la fréquence approche $\frac{1}{6} \approx 0{,}167$.

Rappel de cours. `randint(a, b)` renvoie un entier entre a et b INCLUS (contrairement à `range`).

Erreur fréquente. Écrire `if randint(1, 6) = 6` : erreur de syntaxe. Le test d'égalité s'écrit avec un double signe égal.

Correction de l'exercice 14 – Construire une liste de termes

`append` ajoute un élément à la fin d'une liste.

```
def termes():
    L = [500]
    u = 500
for k in range(7):
    u = 0.9 * u + 30
    L.append(u)
    return L
```

La liste contient déjà u_0 au départ, et la boucle ajoute 7 termes : au total 8 termes, de u_0 à u_7 ✔.

$u_1 = 0{,}9 \times 500 + 30 = 450 + 30 = 480$.

$u_2 = 0{,}9 \times 480 + 30 = 432 + 30 = 462$.

$u_3 = 0{,}9 \times 462 + 30 = 415{,}8 + 30 = 445{,}8$.

Les termes décroissent : 500, 480, 462, 445,8, … mais de moins en moins vite (écarts -20, -18, $-16{,}2$).

Valeur d'équilibre : on cherche ℓ tel que $\ell = 0{,}9\ell + 30$, soit $0{,}1\ell = 30$ et $\ell = 300$.

La suite décroît donc vers 300 sans l'atteindre. Contrôle : $u_7 \approx 404{,}5$, encore loin, mais $u_{30} \approx 308{,}5$.

Réponse. Liste initialisée à `[500]` puis 7 `append` · $u_1 = 480$, $u_2 = 462$, $u_3 = 445{,}8$ · la suite tend vers 300.

Rappel de cours. `L.append(u)` ajoute `u` en fin de liste. Attention au nombre de tours si le premier terme est déjà dans la liste.

Erreur fréquente. Boucler 8 fois alors que u_0 est déjà présent : la liste contiendrait 9 termes, donc un terme de trop.

Correction de l'exercice 15 – Chercher un minimum par balayage

On combine un balayage (boucle `for` sur un compteur entier) et une recherche de minimum (variable « championne »).

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
def f(x):
    return x**2 - 6*x + 11
def minimum():
    xm = 0
    m = f(0)
    for k in range(61):
        x = 0.1 * k
        if f(x) < m:
            m = f(x)
            xm = x
    return xm, m
```

Le `range(61)` couvre $x = 0$ à $x = 6$ par pas de 0,1, soit 61 valeurs (bornes comprises).

Il faut mémoriser DEUX informations : la plus petite valeur `m` et l'abscisse `xm` où elle est atteinte. Les deux se mettent à jour ensemble.

Résultat : le programme renvoie `xm` = 3,0 et `m` = 2,0.

Vérification par le calcul : $f(x) = x^2 - 6x + 11 = (x-3)^2 + 2$, forme canonique qui donne un minimum de 2 en $x = 3$ ✔.

Le balayage a trouvé la valeur EXACTE ici, parce que 3 est un multiple du pas 0,1. Avec un minimum en $x = 3{,}05$, le balayage aurait renvoyé 3,0 ou 3,1 : la précision du résultat est limitée par le pas.

Remarque technique : `f(x)` est appelée deux fois par tour (dans le test et dans l'affectation). On peut l'éviter en stockant `y = f(x)` au début de la boucle — utile si le calcul est coûteux.

Réponse. Balayage `for k in range(61)` avec `x = 0.1*k` et mise à jour de `m` et `xm` · le programme renvoie (3,0 ; 2,0) · confirmé par $f(x) = (x-3)^2 + 2$.

Rappel de cours. Recherche de minimum : on mémorise la plus petite valeur ET son abscisse, toutes deux mises à jour dans le même `if`.

Erreur fréquente. Mettre à jour `m` sans `xm` : on connaîtrait la valeur du minimum sans savoir où il est atteint.

Correction de l'exercice 16 – Type bac : simulation de Monte-Carlo

Une pièce est non conforme si `random() < 0.12`, ce qui se produit avec une probabilité 0,12.

```
from random import random
def echantillon():
    c = 0
    for k in range(50):
        if random() < 0.12:
            c = c + 1
    return c
```

Estimation de la probabilité sur N échantillons :

```
def estimation(N):
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
    c = 0
for k in range(N):
if echantillon() >= 10:
        c = c + 1
return c / N
```

Il y a donc DEUX boucles imbriquées : la boucle interne fabrique un échantillon (50 pièces), la boucle externe répète l'expérience N fois. Avec $N = 10\,000$, le programme effectue 500 000 tirages.

Nombre moyen de pièces non conformes par échantillon : $50 \times 0{,}12 = 6$ pièces.

« Au moins 10 » est donc un évènement plutôt rare : il faut dépasser de 4 pièces la valeur attendue. La simulation renvoie en pratique une valeur voisine de 0,07.

Intérêt de la méthode : le calcul exact de cette probabilité exigerait la loi binomiale (Terminale). La simulation, elle, donne une estimation utilisable dès la première, sans autre outil qu'un compteur.

Réponse. Deux boucles imbriquées : 50 tirages `random() < 0.12` par échantillon, répétés N fois · on attend $50 \times 0{,}12 = 6$ pièces non conformes en moyenne, donc « au moins 10 » est rare.

Rappel de cours. Monte-Carlo : on répète un grand nombre de fois une expérience simulée, et la FRÉQUENCE observée estime la probabilité cherchée.

Erreur fréquente. Appeler `echantillon()` deux fois dans le test (une fois pour comparer, une fois pour compter) : on simulerait deux échantillons différents et le résultat serait faux.

Correction de l'exercice 17 – Type bac : seuil de rentabilité par programme

Bénéfice : $B(q) = 60q - C(q) = 60q - 0{,}5q^2 - 20q - 800 = -0{,}5q^2 + 40q - 800$.

```
def B(q):
return -0.5*q**2 + 40*q - 800
```

Recherche du seuil par boucle `while` :

```
def seuil():
    q = 0
while B(q) <= 0:
    q = q + 1
return q
```

Valeurs calculées : $B(0) = -800$; $B(10) = -50 + 400 - 800 = -450$;
$B(20) = -200 + 800 - 800 = -200$.

$B(30) = -450 + 1\,200 - 800 = -50$: encore négatif.

$B(35) = -612{,}5 + 1\,400 - 800 = -12{,}5$: toujours négatif.

$B(38) = -722 + 1\,520 - 800 = -2$: négatif de peu.

$B(39) = -760{,}5 + 1\,560 - 800 = -0{,}5$: encore négatif !

$B(40) = -800 + 1\,600 - 800 = 0$: nul, donc le test `B(q) <= 0` est encore vrai, la boucle continue.

$B(41) = -840{,}5 + 1\,640 - 800 = -0{,}5$: négatif de nouveau.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Ce comportement est révélateur : le maximum du bénéfice est atteint en $q = 40$ (sommet de la parabole) et il vaut 0. Le bénéfice n'est donc JAMAIS strictement positif.

Conséquence pour le programme : la boucle `while` ne s'arrête jamais — c'est une boucle infinie. Il faut donc modifier l'énoncé du problème : au mieux, l'atelier atteint l'équilibre à 40 pièces, et il ne peut pas dégager de bénéfice avec ce prix de vente.

Correction possible du programme : ajouter une borne de sécurité, `while B(q) <= 0 and q < 1000`, et tester la valeur renvoyée avant de conclure. Tout programme à boucle `while` doit prévoir ce garde-fou.

Réponse. $B(q) = -0{,}5q^2 + 40q - 800$, maximal en $q = 40$ où il vaut 0 : le bénéfice n'est jamais strictement positif et la boucle `while` tournerait indéfiniment · il faut borner la boucle (`and q < 1000`) et conclure à l'équilibre à 40 pièces.

Rappel de cours. Une boucle `while` peut ne jamais s'arrêter si la condition reste vraie : on prévoit toujours une borne de sécurité.

Erreur fréquente. Faire confiance au programme sans vérifier que la solution existe. Ici l'étude mathématique (sommet de la parabole) est indispensable AVANT de lancer la boucle.

Correction de l'exercice 18 – Type bac : déboguer un programme

Déroulons le programme sur `[150, 80, 200]`.

Premier tour, $x = 150$: `c` est remis à 0, puis $150 > 100$ donc `c` = 1. Ensuite `return c` est exécuté : la fonction RENVOIE 1 et s'arrête.

Le programme renvoie donc 1, sans même regarder les valeurs 80 et 200.

Première erreur : `c = 0` est À L'INTÉRIEUR de la boucle. Le compteur est réinitialisé à chaque tour, il ne peut donc jamais dépasser 1.

Seconde erreur : `return c` est aussi dans la boucle. Le `return` interrompt immédiatement la fonction, dès le premier tour.

Programme corrigé : les deux lignes doivent sortir de la boucle, l'une AVANT, l'autre APRÈS.

```
def compte(L):
    c = 0
    for x in L:
        if x > 100:
            c = c + 1
    return c
```

Vérification : $x = 150 \to$ `c` = 1 ; $x = 80 \to$ pas de changement ; $x = 200 \to$ `c` = 2. La fonction renvoie 2 ✔, ce qui est bien le nombre de valeurs supérieures à 100.

Méthode générale : en Python, la STRUCTURE est portée par l'indentation. Un décalage de quatre espaces change complètement le sens du programme — c'est la source d'erreur la plus fréquente au bac.

Réponse. Le programme renvoie 1 · erreurs : `c = 0` et `return c` sont dans la boucle · corrigé, il renvoie 2.

Rappel de cours. Initialiser AVANT la boucle, renvoyer APRÈS la boucle. En Python, c'est l'indentation qui le dit.

Erreur fréquente. Ne corriger qu'une des deux erreurs : en sortant seulement `return`, le compteur resterait remis à zéro et la fonction renverrait 0 ou 1.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

Correction de l’exercice 19 – Type bac : programme de statistique à deux variables

Le point moyen a pour coordonnées les deux moyennes.

```
def pointMoyen(X, Y):
    sx = 0
    sy = 0
    for i in range(len(X)):
        sx = sx + X[i]
        sy = sy + Y[i]
    n = len(X)
    return sx/n, sy/n
```

Ici on boucle sur les INDICES (`range(len(X))`) et non sur les valeurs, car il faut avancer dans les deux listes en parallèle.

Application : `sx` = 1 + 2 + 3 + 4 + 5 = 15 et `sy` = 22 + 41 + 58 + 79 + 100 = 300, avec $n = 5$.

La fonction renvoie $\left(\frac{15}{5} ; \frac{300}{5}\right) = (3 ; 60)$: le point moyen est $G(3 ; 60)$ ✔.

Somme des carrés des résidus pour une droite donnée :

```
def residus(X, Y, a, b):
    S = 0
    for i in range(len(X)):
        r = Y[i] - (a*X[i] + b)
        S = S + r**2
    return S
```

Avec l’ajustement $a = 19{,}4$ et $b = 1{,}8$, les résidus valent +0,8 ; +0,4 ; −2 ; −0,4 ; +1,2, donc `S` = 0,64 + 0,16 + 4 + 0,16 + 1,44 = 6,4.

Avec la droite des points extrêmes $a = 19{,}5$ et $b = 2{,}5$, on obtiendrait `S` = 11,5 : plus grand, donc moins bon ajustement.

Ce programme permet ainsi de COMPARER numériquement deux droites — c’est exactement le critère des moindres carrés.

Réponse. Point moyen : deux accumulateurs, `return sx/n, sy/n`, soit $G(3 ; 60)$ · résidus : `r = Y[i] - (a*X[i] + b)` et `S = S + r**2`, qui donne 6,4 pour $y = 19{,}4x + 1{,}8$.

Rappel de cours. Pour parcourir deux listes en parallèle, on boucle sur les INDICES : `for i in range(len(X))`.

Erreur fréquente. Oublier les parenthèses dans `Y[i] - (a*X[i] + b)` : sans elles, le `b` serait ajouté au lieu d’être retranché, et tous les résidus seraient faux.

Correction de l’exercice 20 – Type bac : observer la fluctuation d’échantillonnage

Simulation d’un échantillon et calcul de sa fréquence :

```
from random import random
```

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s’expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).

```
def frequence(n):
    c = 0
    for k in range(n):
        if random() < 0.2:
            c = c + 1
    return c / n
```

Écart maximal sur 100 échantillons : on garde le plus grand écart rencontré, en valeur absolue.

```
def ecartMax(n):
    m = 0
    for k in range(100):
        e = abs(frequence(n) - 0.2)
        if e > m:
            m = e
    return m
```

Ici l'initialisation `m = 0` est légitime : un écart en valeur absolue est toujours positif ou nul, donc 0 ne peut pas être dépassé par le bas.

Avec $n = 20$, le programme renvoie typiquement un écart maximal de l'ordre de 0,20 à 0,25 : certains échantillons donnent 0 ou 0,40 de fréquence, très loin de 0,2.

Avec $n = 500$, l'écart maximal tombe à environ 0,05 : toutes les fréquences observées restent dans $[0{,}15\,;\,0{,}25]$.

Conclusion : la fluctuation d'échantillonnage DIMINUE quand la taille de l'échantillon augmente. En multipliant n par 25 (de 20 à 500), on divise l'amplitude des écarts par environ 5 — c'est-à-dire par $\sqrt{25}$.

Conséquence pratique en contrôle qualité : un échantillon de 20 pièces ne permet de conclure à peu près rien sur la proportion réelle ; il faut des centaines de pièces pour l'estimer sérieusement.

Réponse. `frequence(n)` compte les `random() < 0.2` puis divise par n · `ecartMax(n)` garde le plus grand `abs(f - 0.2)` sur 100 échantillons · l'écart passe d'environ 0,2 pour $n = 20$ à 0,05 pour $n = 500$.

Rappel de cours. La fluctuation d'échantillonnage décroît quand n croît : multiplier n par 4 divise environ par 2 l'amplitude des écarts.

Erreur fréquente. Comparer des écarts obtenus sur des NOMBRES de pièces plutôt que sur des FRÉQUENCES : 4 pièces d'écart sur 20 et sur 500 n'ont pas du tout la même signification.

mathlvl.fr — Tous droits réservés. Toute reproduction, diffusion, revente ou exploitation commerciale, même partielle, est strictement interdite.
Tout contrevenant s'expose à des poursuites en vertu du Code de la propriété intellectuelle (art. L.335-2 et suivants).