# Algorithmique et Python — Terminale STI2D

## 20 exercices corrigés · mathlvl.fr

Énoncés et corrigés détaillés

### Présentation

En terminale STI2D, Python sert à faire trois choses que le calcul à la main ne permet pas : CUMULER (la somme des termes d'une suite, le triangle de Pascal ligne après ligne), CHERCHER UN SEUIL (à quel rang le cumul dépasse-t-il une cible ?) et SIMULER (méthode de Monte-Carlo, marches aléatoires). Ces 20 exercices corrigés suivent ce programme : sommes et listes, construction du triangle de Pascal, simulation d'une loi binomiale, estimation d'une probabilité ou d'une aire par Monte-Carlo, marche aléatoire. Chaque correction donne la TRACE des variables — la seule façon de vérifier un programme sans machine, et ce que le bac attend. 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/terminale-sti2d/algorithmique-python-terminale-sti2d

### Exercice 1 – Somme des éléments d'une liste

On dispose de la liste des productions journalières `L = [120, 135, 98, 142, 105]`.

**1.** Écris une fonction qui renvoie la somme des éléments d'une liste, sans utiliser `sum`.

**2.** Donne la trace de l'accumulateur sur cette liste.

### Exercice 2 – Construire la liste des termes d'une suite

Soit la suite arithmétique définie par $u_1 = 120$ et $u_{n+1} = u_n + 8$ (production hebdomadaire).

**1.** Écris un programme construisant la liste des 6 premiers termes.

**2.** Donne cette liste.

### Exercice 3 – Somme des premiers entiers

On veut calculer $1 + 2 + \cdots + n$ par programme, puis comparer à la formule $\dfrac{n(n+1)}{2}$.

**1.** Écris la fonction.

**2.** Que renvoie-t-elle pour $n = 100$ ? Vérifie avec la formule.

### Exercice 4 – Lire un programme de marche aléatoire

On exécute le programme suivant, où `randint(0, 1)` renvoie 0 ou 1 au hasard :

```
from random import randint
x = 0
for k in range(4):
if randint(0, 1) == 1:
x = x + 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).

```
        else:
            x = x - 1
```

**1.** Que représente la variable `x` à la fin ?

**2.** Quelles valeurs finales sont possibles ?

## Exercice 5 – Simuler un succès de probabilité p

On rappelle que `random()` renvoie un réel au hasard dans $[0\,;1[$, uniformément.

**1.** Comment simuler un évènement de probabilité 0,12 ?

**2.** Écris une fonction qui renvoie 1 en cas de succès et 0 sinon.

**3.** Comment simuler un évènement de probabilité 0,5 avec `randint` ?

## Exercice 6 – Somme d'une suite géométrique

Un placement verse 2 000 € la première année, puis 6% de plus chaque année. On veut le total versé sur $n$ années.

**1.** Écris une fonction calculant ce total par accumulation.

**2.** Donne la trace pour $n = 4$.

**3.** Compare avec la formule de la somme d'une suite géométrique.

## Exercice 7 – Une ligne du triangle de Pascal

On veut calculer la ligne $n = 5$ du triangle de Pascal à partir de la ligne $n = 4$, soit `[1, 4, 6, 4, 1]`.

**1.** Écris une fonction qui, à partir d'une ligne, renvoie la suivante.

**2.** Applique-la à `[1, 4, 6, 4, 1]`.

## Exercice 8 – Moyenne et maximum d'une série

Une liste contient les relevés de température `T = [62, 68, 71, 65, 74, 69]`.

**1.** Écris une fonction renvoyant la moyenne, sans utiliser `sum`.

**2.** Écris une fonction renvoyant le maximum, sans utiliser `max`.

**3.** Applique-les à cette liste.

## Exercice 9 – Première simulation de Monte-Carlo

Une production comporte 7% de pièces non conformes. On veut estimer par simulation la probabilité qu'un prélèvement de 10 pièces en contienne au moins une.

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

**2.** Écris une fonction estimant la probabilité cherchée sur $N$ simulations.

**3.** Quelle valeur théorique la simulation doit-elle approcher ?

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 10 – Cumul jusqu'à un seuil

Une ligne produit 500 pièces le premier mois, puis 4% de plus chaque mois. On veut savoir au bout de combien de mois la production CUMULÉE dépasse 10 000 pièces.

**1.** Écris le programme.

**2.** Donne la trace des quatre premiers mois.

**3.** Estime le résultat.

## Exercice 11 – Triangle de Pascal complet

On veut afficher les $n$ premières lignes du triangle de Pascal, en les stockant dans une liste de listes.

**1.** Écris la fonction.

**2.** Que renvoie-t-elle pour $n = 5$ ?

**3.** Comment en déduire $\binom{4}{2}$ ?

## Exercice 12 – Calculer un coefficient binomial

On veut calculer $\binom{n}{k}$ sans construire tout le triangle, en utilisant la relation $\binom{n}{k} = \binom{n}{k-1} \times \frac{n-k+1}{k}$.

**1.** Écris la fonction.

**2.** Donne la trace pour $\binom{6}{3}$.

**3.** Compare avec la lecture sur le triangle.

## Exercice 13 – Simuler une loi binomiale

On veut estimer par simulation $P(X = 2)$ où $X$ suit $B(5\,;0{,}2)$.

**1.** Écris une fonction simulant une réalisation de $X$.

**2.** Écris une fonction estimant $P(X = 2)$ sur $N$ simulations.

**3.** Compare à la valeur théorique.

## Exercice 14 – Marche aléatoire et position finale

Un chariot autoguidé fait $n$ déplacements élémentaires, chacun de $+1$ ou $-1$ mètre avec la même probabilité.

**1.** Écris une fonction simulant la position finale après $n$ pas.

**2.** Écris une fonction estimant la position moyenne sur $N$ simulations.

**3.** Que doit donner cette moyenne ? Et la DISTANCE moyenne à l'origine ?

## Exercice 15 – Monte-Carlo pour estimer une aire

On veut estimer l'aire du quart de disque de rayon 1 inclus dans le carré $[0\,;1] \times [0\,;1]$, en tirant des points au hasard dans ce carré.

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).

**1.** Quelle condition un point $(x\,;y)$ doit-il vérifier pour être dans le quart de disque ?

**2.** Écris le programme d'estimation sur $N$ tirages.

**3.** Quelle valeur doit-on obtenir, et comment en déduire une estimation de $\pi$ ?

## Exercice 16 – Type bac : marche aléatoire d'un stock

Un stock contient 50 pièces. Chaque jour, il augmente de 3 pièces avec une probabilité 0,5, ou diminue de 4 pièces avec une probabilité 0,5. On veut savoir s'il risque de tomber à zéro en 30 jours.

**1.** Écris une fonction simulant l'évolution sur 30 jours et renvoyant 1 si le stock s'annule (ou devient négatif), 0 sinon.

**2.** Écris une fonction estimant la probabilité de rupture sur $N$ simulations.

**3.** Calcule la variation moyenne journalière et prévois la tendance.

## Exercice 17 – Type bac : simuler un plan de contrôle

Un plan de contrôle prélève 20 pièces et refuse le lot dès 3 pièces non conformes. On veut comparer, par simulation, son comportement sur un lot à 2% et sur un lot à 8% de non conformes.

**1.** Écris une fonction simulant un prélèvement dans un lot de proportion $p$ et renvoyant 1 si le lot est refusé.

**2.** Écris une fonction estimant la probabilité de refus sur $N$ simulations.

**3.** Quelles valeurs théoriques la simulation doit-elle approcher ?

**4.** Conclus sur la qualité du plan.

## Exercice 18 – Type bac : déboguer un calcul de somme

Un élève veut calculer la somme $S = u_1 + u_2 + \cdots + u_{10}$ de la suite géométrique $u_n = 3 \times 2^{n-1}$. Il écrit :

```
def somme():
    u = 3
for n in range(10):
    S = 0
    S = S + u
    u = 2 * u
    return S
```

**1.** Que renvoie ce programme ?

**2.** Identifie l'erreur et corrige.

**3.** Donne la valeur exacte de $S$ et vérifie par la formule.

## Exercice 19 – Type bac : aire sous une courbe par Monte-Carlo

On veut estimer l'aire du domaine situé sous la courbe de $f(x) = x^2$ entre $x = 0$ et $x = 2$, sachant que $f$ est majorée par 4 sur cet intervalle.

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).

**1.** Dans quel rectangle doit-on tirer les points ? Quelle est son aire ?

**2.** Écris le programme d'estimation.

**3.** Quelle valeur doit-on obtenir ?

**4.** Comment améliorer la précision ?

## Exercice 20 – Type bac : cumul, seuil et liste

Une installation photovoltaïque produit 1 200 kWh le premier mois, puis 2% de moins chaque mois (vieillissement et saison). On veut la production cumulée et le mois où elle dépasse 20 000 kWh.

**1.** Écris une fonction renvoyant la liste des productions mensuelles sur $n$ mois.

**2.** Écris une fonction renvoyant le mois où le cumul dépasse un seuil $s$.

**3.** Détermine ce mois par le calcul.

**4.** Quelle est la production cumulée maximale possible, même sur un temps infini ?

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 – Somme des éléments d'une liste

On utilise un ACCUMULATEUR : une variable initialisée à 0, à laquelle on ajoute chaque élément.

```
def somme(L):
    S = 0
    for x in L:
        S = S + x
    return S
```

Trace : $S = 0$, puis 120, puis 255, puis 353, puis 495, puis 600.

La fonction renvoie 600 : la production totale des cinq jours.

Deux points de vigilance : `S = 0` doit être AVANT la boucle, et `return S` APRÈS.

**Réponse.** Accumulateur `S = S + x` dans la boucle · trace 120, 255, 353, 495, 600 · la fonction renvoie 600.

**Rappel de cours.** Somme d'une liste : accumulateur initialisé à 0 avant la boucle, renvoyé après.

**Erreur fréquente.** Placer `return S` dans la boucle : la fonction s'arrêterait au premier élément et renverrait 120.

## Correction de l'exercice 2 – Construire la liste des termes d'une suite

On part d'une liste contenant le premier terme et on ajoute les suivants avec `append`.

```
def termes():
    L = [120]
    u = 120
    for k in range(5):
        u = u + 8
        L.append(u)
    return L
```

La liste contient déjà $u_1$, et la boucle ajoute 5 termes : au total 6 termes ✔.

Résultat : `[120, 128, 136, 144, 152, 160]`.

Contrôle par la formule : $u_6 = 120 + 5 \times 8 = 160$ ✔.

**Réponse.** `L = [120]` puis 5 `append` · liste `[120, 128, 136, 144, 152, 160]`.

**Rappel de cours.** `L.append(u)` ajoute `u` en fin de liste. Attention au nombre de tours quand le premier terme y est déjà.

**Erreur fréquente.** Boucler 6 fois : la liste contiendrait 7 termes, soit un de trop.

## Correction de l'exercice 3 – Somme des premiers entiers

```
def somme(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).

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

Le `range(1, n+1)` parcourt 1, 2, …, $n$ : la borne de droite est EXCLUE, d'où le +1.

Pour $n = 100$, la fonction renvoie 5 050.

Vérification par la formule : $\frac{100 \times 101}{2} = 5\,050$ ✔.

Comparaison des deux approches : la formule est instantanée, la boucle effectue 100 additions. Sur $n = 10^9$, seule la formule reste utilisable.

**Réponse.** Boucle `for k in range(1, n+1)` avec accumulateur · pour $n = 100$, elle renvoie $5\,050 = \frac{100 \times 101}{2}$ ✔.

**Rappel de cours.** `range(a, b)` s'arrête à $b - 1$ : pour aller jusqu'à $n$, on écrit `range(1, n+1)`.

**Erreur fréquente.** Écrire `range(1, n)` : la somme s'arrêterait à $n - 1$ et vaudrait 4 950 pour $n = 100$.

## Correction de l'exercice 4 – Lire un programme de marche aléatoire

À chaque tour, x avance de +1 ou recule de −1, avec la même probabilité. C'est une MARCHE ALÉATOIRE sur une droite.

À la fin, x représente la position atteinte après 4 pas, en partant de 0.

Si $k$ pas sont vers la droite et $4 - k$ vers la gauche, alors $x = k - (4 - k) = 2k - 4$.

Pour $k = 0$ à 4, les positions possibles sont −4, −2, 0, 2, 4.

Les positions IMPAIRES sont impossibles : avec un nombre pair de pas, la position finale est toujours paire.

Remarque : le nombre $k$ de pas vers la droite suit une loi binomiale $B(4\,;0{,}5)$, de répartition 1-4-6-4-1 sur 16. La position 0 est donc la plus probable ($\frac{6}{16}$).

**Réponse.** x est la position après 4 pas · valeurs possibles −4, −2, 0, 2, 4 (jamais impaires), la plus probable étant 0 avec $\frac{6}{16}$.

**Rappel de cours.** Marche aléatoire : à chaque pas ±1. Après $n$ pas, la position vaut $2k - n$ où $k$ est le nombre de pas positifs.

**Erreur fréquente.** Croire que toutes les positions entre −4 et 4 sont possibles : seules les valeurs de même parité que $n$ le sont.

## Correction de l'exercice 5 – Simuler un succès de probabilité p

Le tirage étant uniforme sur $[0\,;1[$, la probabilité que `random()` soit inférieur à 0,12 vaut exactement 0,12.

On teste donc `random() < 0.12`.

```
from random import random

def essai():
    if random() < 0.12:
        return 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).

```
return 0
```

Le `return 0` final n'a pas besoin de `else` : il n'est atteint que si le test a échoué.

Pour une probabilité 0,5, on peut aussi utiliser `randint(0, 1) == 1`, qui donne 0 ou 1 de façon équiprobable.

À retenir : `random()` sert pour une probabilité quelconque, `randint` pour des issues entières équiprobables.

**Réponse.** Tester `random() < 0.12` · fonction renvoyant 1 ou 0 · et `randint(0, 1) == 1` pour une probabilité 0,5.

**Rappel de cours.** `random() < p` simule un succès de probabilité $p$ · `randint(a, b)` renvoie un entier de $a$ à $b$ INCLUS.

**Erreur fréquente.** Écrire `random() < 12` : la condition serait toujours vraie, puisque `random()` renvoie un nombre inférieur à 1.

## Correction de l'exercice 6 – Somme d'une suite géométrique

On accumule les versements, en multipliant le versement courant par 1,06 à chaque tour.

```
def total(n):
    v = 2000
    S = 0
    for k in range(n):
        S = S + v
        v = 1.06 * v
    return S
```

L'ordre des deux lignes compte : on ajoute d'abord le versement de l'année, PUIS on le fait croître pour l'année suivante.

Trace pour $n = 4$ : $S = 2\,000$ puis $v = 2\,120$ ; $S = 4\,120$ puis $v = 2\,247{,}2$ ; $S = 6\,367{,}2$ puis $v = 2\,382{,}03$ ; $S = 8\,749{,}23$.

La fonction renvoie environ 8 749,23 €.

Contrôle par la formule : $S = 2\,000 \times \dfrac{1 - 1{,}06^4}{1 - 1{,}06}$. Or $1{,}06^4 \approx 1{,}262\,477$, donc $S \approx 2\,000 \times \dfrac{-0{,}262\,477}{-0{,}06} \approx 8\,749{,}2$ € ✔.

**Réponse.** Boucle avec `S = S + v` puis `v = 1.06*v` · trace 2 000 ; 4 120 ; 6 367,2 ; 8 749,23 · conforme à la formule $2\,000\dfrac{1 - 1{,}06^4}{1 - 1{,}06}$ ✔.

**Rappel de cours.** Deux variables : `v` pour le terme courant, `S` pour le cumul. On ajoute avant de faire évoluer le terme.

**Erreur fréquente.** Inverser les deux lignes : on ajouterait le versement de l'année suivante dès le premier tour, et le total serait trop élevé de 6%.

## Correction de l'exercice 7 – Une ligne du triangle de Pascal

Principe : la ligne suivante commence et finit par 1, et chaque terme intérieur est la somme de deux termes consécutifs de la ligne précédente.

```
def suivante(L):
    M = [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).

```
    for i in range(len(L) - 1):
        M.append(L[i] + L[i+1])
    M.append(1)
    return M
```

La boucle tourne `len(L) - 1` fois, soit 4 fois pour une ligne de 5 termes : cela produit les 4 termes intérieurs.

Application : $L[0] + L[1] = 1 + 4 = 5$ ; $L[1] + L[2] = 4 + 6 = 10$ ; $L[2] + L[3] = 6 + 4 = 10$ ; $L[3] + L[4] = 4 + 1 = 5$.

Avec les deux 1 ajoutés : `[1, 5, 10, 10, 5, 1]` ✔ C'est bien la ligne $n = 5$.

Contrôle : la ligne obtenue compte 6 termes, soit un de plus que la précédente ✔ et sa somme vaut $32 = 2^5$ ✔.

**Réponse.** Fonction ajoutant 1, puis les sommes `L[i] + L[i+1]`, puis 1 · appliquée à `[1,4,6,4,1]`, elle renvoie `[1,5,10,10,5,1]`.

**Rappel de cours.** $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ : c'est exactement ce que fait `L[i] + L[i+1]`.

**Erreur fréquente.** Boucler `range(len(L))` : à la dernière itération, `L[i+1]` sortirait de la liste et Python renverrait une erreur d'indice.

## Correction de l'exercice 8 – Moyenne et maximum d'une série

Moyenne : on accumule puis on divise APRÈS la boucle.

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

Maximum : on retient une valeur « championne », initialisée au premier élément.

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

Application : somme = 62 + 68 + 71 + 65 + 74 + 69 = 409, donc moyenne = $\frac{409}{6} \approx 68{,}17$ °C.

Maximum : la championne passe de 62 à 68, puis 71, puis 74 — et reste à 74. La fonction renvoie 74 °C.

Pourquoi initialiser à `T[0]` et non à 0 ? Parce qu'avec des relevés négatifs (des écarts, par exemple), l'initialisation à 0 renverrait 0, qui n'est pas dans la liste.

**Réponse.** Moyenne : accumulateur puis `return S / len(T)`, soit ≈ 68,17 °C · maximum : champion initialisé à `T[0]`, soit 74 °C.

**Rappel de cours.** Accumuler DANS la boucle, diviser APRÈS · maximum : initialiser au

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).

premier élément de la liste.

**Erreur fréquente.** Diviser par `len(T)` à chaque tour : on calculerait des moyennes partielles sans signification.

## Correction de l'exercice 9 – Première simulation de Monte-Carlo

```
from random import random
def prelevement():
    c = 0
    for k in range(10):
        if random() < 0.07:
            c = c + 1
    return c
def estimation(N):
    c = 0
    for k in range(N):
        if prelevement() >= 1:
            c = c + 1
    return c / N
```

Valeur théorique : le nombre de non conformes suit $B(10\,;0{,}07)$, donc $P(X \geq 1) = 1 - 0{,}93^{10}$.

$0{,}93^{10} \approx 0{,}483\,98$, donc $P(X \geq 1) \approx 0{,}516$.

La simulation doit donc renvoyer une valeur proche de 0,52. Avec $N = 100$, on obtiendra typiquement entre 0,42 et 0,62 ; avec $N = 100\,000$, on sera à 0,005 près.

Deux exécutions donnent des résultats DIFFÉRENTS : c'est normal pour une simulation, ce n'est pas un défaut du programme.

**Réponse.** Deux boucles imbriquées (10 pièces, $N$ prélèvements) · la simulation doit approcher $1 - 0{,}93^{10} \approx 0{,}516$.

**Rappel de cours.** Monte-Carlo : on répète l'expérience un grand nombre de fois, la FRÉQUENCE observée estime la probabilité.

**Erreur fréquente.** Appeler `prelevement()` deux fois dans une même condition : on simulerait deux prélèvements différents et le résultat serait faux.

## Correction de l'exercice 10 – Cumul jusqu'à un seuil

Deux variables à faire évoluer : la production du mois et le cumul.

```
def mois():
    p = 500
    S = 0
    n = 0
    while S <= 10000:
        S = S + p
        p = 1.04 * p
```

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).

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

Trace : $n = 1$ donne $S = 500$ et $p = 520$ ; $n = 2$ donne $S = 1\,020$ et $p = 540{,}8$ ; $n = 3$ donne $S = 1\,560{,}8$ et $p \approx 562{,}43$ ; $n = 4$ donne $S \approx 2\,123{,}23$.

Le cumul croît d'environ 500 à 600 par mois : il faudra donc une quinzaine de mois.

Calcul par la formule : $S_n = 500 \times \dfrac{1 - 1{,}04^{\,n}}{1 - 1{,}04} = 12\,500\left(1{,}04^{\,n} - 1\right)$.

Condition $S_n > 10\,000$ : $1{,}04^{\,n} > 1{,}8$, donc $n > \dfrac{\log 1{,}8}{\log 1{,}04} \approx 14{,}99$.

Le programme renvoie donc 15 : le cumul dépasse 10 000 pièces au 15e mois.

Contrôle : $S_{15} = 12\,500(1{,}04^{15} - 1) \approx 12\,500 \times 0{,}800\,94 \approx 10\,011{,}8$ ✔ et $S_{14} \approx 9\,146$ ✔.

**Réponse.** Boucle `while S <= 10000` avec `S = S + p` puis `p = 1.04*p` · trace 500 ; 1 020 ; 1 560,8 ; 2 123,23 · le programme renvoie 15 mois.

**Rappel de cours.** Cumul avec seuil : boucle `while`, deux variables (terme courant et cumul) et un compteur.

**Erreur fréquente.** Oublier `n = n + 1` : la boucle s'exécuterait correctement mais renverrait 0 — ou tournerait indéfiniment si l'on oubliait aussi la mise à jour de `S`.

## Correction de l'exercice 11 – Triangle de Pascal complet

On réutilise le principe de la ligne suivante, en mémorisant toutes les lignes.

```
def pascal(n):
    T = [[1]]
    for i in range(1, n):
        L = T[i-1]
        M = [1]
        for j in range(len(L) - 1):
            M.append(L[j] + L[j+1])
        M.append(1)
        T.append(M)
    return T
```

Pour $n = 5$, la fonction renvoie `[[1], [1,1], [1,2,1], [1,3,3,1], [1,4,6,4,1]]`.

Les lignes correspondent à $n = 0$ à $n = 4$ : la ligne d'indice $i$ contient les coefficients $\binom{i}{k}$.

Pour $\binom{4}{2}$ : c'est le terme d'indice 2 de la ligne d'indice 4, soit `T[4][2]` $= 6$ ✔.

Attention aux indices : `pascal(5)` donne les lignes 0 à 4. Pour disposer de $\binom{n}{k}$, il faut donc appeler `pascal(n+1)`.

Contrôle : `T[4]` a 5 éléments, de somme $16 = 2^4$ ✔.

**Réponse.** Boucle imbriquée construisant chaque ligne depuis la précédente · `pascal(5)` renvoie les lignes $n = 0$ à $4$ · $\binom{4}{2} =$ `T[4][2]` $= 6$.

**Rappel de cours.** Une liste de listes s'indexe deux fois : `T[i][j]` est le terme $j$ de la

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).

ligne $i$, et vaut $\binom{i}{j}$.

**Erreur fréquente.** Appeler `pascal(4)` pour obtenir $\binom{4}{2}$ : la dernière ligne serait celle de $n = 3$, et `T[4]` provoquerait une erreur d'indice.

## Correction de l'exercice 12 – Calculer un coefficient binomial

On part de $\binom{n}{0} = 1$ et l'on applique la relation $k$ fois.

```
def coeff(n, k):
    c = 1
    for i in range(1, k+1):
        c = c * (n - i + 1) / i
    return c
```

Trace pour $n = 6$ et $k = 3$ : $c = 1$.

$i = 1$ : $c = 1 \times \frac{6}{1} = 6$.

$i = 2$ : $c = 6 \times \frac{5}{2} = 15$.

$i = 3$ : $c = 15 \times \frac{4}{3} = 20$.

La fonction renvoie 20.

Vérification sur le triangle : la ligne $n = 6$ est 1 6 15 20 15 6 1, et son terme d'indice 3 vaut bien 20 ✔.

Avantage de cette méthode : elle ne demande que $k$ multiplications, là où le triangle complet exige de construire $n$ lignes. Pour $\binom{50}{2}$, c'est deux opérations contre cinquante lignes.

**Réponse.** Boucle `c = c * (n - i + 1) / i` · trace $1 \to 6 \to 15 \to 20$ pour $\binom{6}{3}$ · conforme au triangle ✔.

**Rappel de cours.** $\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{1 \times 2 \times \cdots \times k}$ : on alterne multiplications et divisions pour rester sur de petits nombres.

**Erreur fréquente.** Faire toutes les multiplications avant les divisions : sur de grands $n$, les produits deviennent énormes et les erreurs d'arrondi apparaissent.

## Correction de l'exercice 13 – Simuler une loi binomiale

Une réalisation de $X$ s'obtient en comptant les succès sur 5 épreuves de probabilité 0,2.

```
from random import random
def realisation():
    c = 0
    for k in range(5):
        if random() < 0.2:
            c = c + 1
    return c
```

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 estime(N):
    c = 0
    for k in range(N):
        if realisation() == 2:
            c = c + 1
    return c / N
```

Noter le test `== 2` (égalité stricte) et non `>= 2` : on veut $P(X = 2)$, pas $P(X \geq 2)$.

Valeur théorique : $P(X = 2) = \binom{5}{2} \times 0{,}2^2 \times 0{,}8^3 = 10 \times 0{,}04 \times 0{,}512 = 0{,}204\,8$.

La simulation doit donc renvoyer une valeur proche de 0,205.

Ordre de grandeur de l'écart : avec $N = 1\,000$, on obtient typiquement entre 0,18 et 0,23 ; avec $N = 1\,000\,000$, on est à 0,001 près.

**Réponse.** Comptage de 5 épreuves à 0,2, puis fréquence des réalisations égales à 2 · valeur théorique $P(X = 2) = 0{,}204\,8$.

**Rappel de cours.** Simuler une loi binomiale : on répète $n$ épreuves de Bernoulli et l'on COMPTE les succès.

**Erreur fréquente.** Tester `>= 2` au lieu de `== 2` : on estimerait $P(X \geq 2) \approx 0{,}263$, une autre probabilité.

## Correction de l'exercice 14 – Marche aléatoire et position finale

```
from random import randint
def marche(n):
    x = 0
    for k in range(n):
        if randint(0, 1) == 1:
            x = x + 1
        else:
            x = x - 1
    return x
def moyenne(n, N):
    S = 0
    for k in range(N):
        S = S + marche(n)
    return S / N
```

La position moyenne doit tendre vers 0 : les pas +1 et −1 étant équiprobables, ils se compensent en moyenne.

En revanche la DISTANCE moyenne à l'origine n'est pas nulle : il suffit de remplacer `marche(n)` par `abs(marche(n))` pour l'estimer.

Pour $n = 100$, cette distance moyenne vaut environ 8 mètres — de l'ordre de $\sqrt{n} = 10$, pas de $n = 100$.

C'est le résultat fondamental des marches aléatoires : on s'éloigne comme $\sqrt{n}$, et non

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).

proportionnellement au nombre de pas.

Conséquence pratique : un chariot qui dérive au hasard s'écarte lentement, mais il s'écarte — et l'écart typique quadruple quand le nombre de pas est multiplié par 16.

**Réponse.** Simulation par `randint(0, 1)` et $\pm 1$ · la position moyenne tend vers 0 · la distance moyenne `abs(...)` croît comme $\sqrt{n}$ (environ 8 m pour $n = 100$).

**Rappel de cours.** Marche aléatoire symétrique : position moyenne nulle, mais éloignement typique de l'ordre de $\sqrt{n}$.

**Erreur fréquente.** Conclure « la position moyenne est nulle, donc le chariot reste sur place » : c'est la MOYENNE des positions qui est nulle, pas la position elle-même.

## Correction de l'exercice 15 – Monte-Carlo pour estimer une aire

Le point $(x\,;y)$ est dans le quart de disque si sa distance à l'origine est inférieure à 1, c'est-à-dire si $x^2 + y^2 \leq 1$.

```
from random import random

def aire(N):
    c = 0
    for k in range(N):
        x = random()
        y = random()
        if x**2 + y**2 <= 1:
            c = c + 1
    return c / N
```

Le carré a une aire de 1, donc la proportion de points tombés dans le quart de disque estime directement son AIRE.

Or l'aire d'un quart de disque de rayon 1 vaut $\frac{\pi}{4} \approx 0{,}785\,4$.

La fonction doit donc renvoyer une valeur proche de 0,785.

Estimation de $\pi$ : il suffit de multiplier par 4. Avec $N = 10\,000$, on obtient typiquement $\pi \approx 3{,}13$ à $3{,}15$.

Précision : la méthode de Monte-Carlo converge lentement — pour gagner un chiffre décimal, il faut multiplier $N$ par 100. Elle vaut donc surtout pour les aires qu'on ne sait pas calculer autrement.

Point technique : il faut deux appels distincts à `random()`, un pour `x` et un pour `y`. Réutiliser la même valeur ne tirerait que des points de la diagonale.

**Réponse.** Condition $x^2 + y^2 \leq 1$ · fréquence des points intérieurs sur $N$ tirages · elle approche $\frac{\pi}{4} \approx 0{,}785$, et $4 \times$ cette fréquence estime $\pi$.

**Rappel de cours.** Monte-Carlo pour une aire : proportion de points tombés dans la région × aire du domaine de tirage.

**Erreur fréquente.** Écrire `x = y = random()` : les deux coordonnées seraient égales et tous les points sur la diagonale du carré.

## Correction de l'exercice 16 – Type bac : marche aléatoire d'un stock

```
from random import randint
```

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 rupture():
    s = 50
    for j in range(30):
        if randint(0, 1) == 1:
            s = s + 3
        else:
            s = s - 4
        if s <= 0:
            return 1
    return 0
```

Le test `if s <= 0` est placé À L'INTÉRIEUR de la boucle : une rupture même temporaire doit être détectée, pas seulement l'état final.

```
def proba(N):
    c = 0
    for k in range(N):
        c = c + rupture()
    return c / N
```

Comme `rupture()` renvoie 0 ou 1, on peut l'ajouter directement au compteur : c'est plus court qu'un `if`.

Variation moyenne journalière : $0{,}5 \times 3 + 0{,}5 \times (-4) = 1{,}5 - 2 = -0{,}5$ pièce par jour.

Tendance : le stock baisse en moyenne de 0,5 pièce par jour, soit 15 pièces sur 30 jours. Il devrait donc finir autour de 35 pièces.

Mais la tendance moyenne ne dit pas tout : les fluctuations peuvent faire descendre le stock bien plus bas. C'est justement ce que la simulation permet de chiffrer.

Ordre de grandeur : avec une baisse moyenne de 15 pièces et des fluctuations de l'ordre de $\sqrt{30} \times 3{,}5 \approx 19$ pièces, une rupture est possible mais peu fréquente — la simulation donne environ 3%.

**Réponse.** Test `if s <= 0: return 1` dans la boucle · comptage sur $N$ simulations · variation moyenne $-0{,}5$ pièce/jour, soit $-15$ sur 30 jours (stock final attendu $\approx 35$), avec un risque de rupture d'environ 3%.

**Rappel de cours.** Espérance de la variation $= \sum p_i v_i$. Elle donne la TENDANCE, la simulation donne le RISQUE.

**Erreur fréquente.** Tester la rupture seulement à la fin des 30 jours : un stock passé par $-2$ puis remonté serait compté comme non rompu, alors que la rupture a bien eu lieu.

## Correction de l'exercice 17 – Type bac : simuler un plan de contrôle

```
from random import random
def refus(p):
    c = 0
    for k in range(20):
        if random() < p:
```

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 = c + 1
    if c >= 3:
        return 1
    return 0
```

Le test `c >= 3` est APRÈS la boucle : on compte d'abord les 20 pièces, puis on décide.

```
def estimation(p, N):
    c = 0
    for k in range(N):
        c = c + refus(p)
    return c / N
```

Valeurs théoriques, par la loi binomiale $B(20\,;p)$ :

Pour $p = 0{,}02$ : $P(X \geq 3) \approx 0{,}007$ — le programme doit renvoyer environ 0,007.

Pour $p = 0{,}08$ : $P(X \geq 3) \approx 0{,}212$ — le programme doit renvoyer environ 0,21.

Conclusion : le plan refuse à tort moins de 1% des bons lots (protection du fournisseur), mais il ne détecte qu'un lot médiocre sur cinq (protection insuffisante du client).

Remarque sur la simulation : pour estimer une probabilité de 0,007, il faut un grand $N$. Avec $N = 1\,000$, on n'observerait que 7 refus en moyenne, ce qui est trop peu pour une estimation stable — il faut $N = 100\,000$ au moins.

C'est une limite générale de Monte-Carlo : plus l'évènement est rare, plus il faut de simulations.

**Réponse.** Comptage sur 20 pièces puis test `c >= 3` · la simulation doit approcher 0,007 pour $p = 0{,}02$ et 0,212 pour $p = 0{,}08$ · plan peu sévère pour le client.

**Rappel de cours.** Une simulation valide un calcul théorique : les deux doivent concorder, sinon l'un des deux est faux.

**Erreur fréquente.** Placer le test `if c >= 3` dans la boucle : le prélèvement s'arrêterait avant les 20 pièces, ce qui n'est pas le plan décrit (même si le verdict final serait identique ici).

## Correction de l'exercice 18 – Type bac : déboguer un calcul de somme

Déroulons : à chaque tour, `S` est REMIS À ZÉRO, puis reçoit `u`.

Au dernier tour ($n = 9$), `u` vaut $3 \times 2^9 = 1\,536$, donc `S` = 1 536.

Le programme renvoie donc 1 536 : c'est le DERNIER terme, pas la somme.

Erreur : `S = 0` est à l'intérieur de la boucle. Il doit être AVANT, pour n'être exécuté qu'une fois.

Programme corrigé :

```
def somme():
    u = 3
    S = 0
    for n in range(10):
        S = S + u
        u = 2 * u
    return S
```

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).

Valeur exacte : $S = 3 + 6 + 12 + \cdots + 1\,536$, somme de 10 termes géométriques de raison 2.

$$S = 3 \times \frac{1 - 2^{10}}{1 - 2} = 3 \times \frac{1 - 1\,024}{-1} = 3 \times 1\,023 = 3\,069.$$

Le programme corrigé renvoie bien 3 069.

Contrôle : la somme doit être légèrement inférieure au double du dernier terme ($2 \times 1\,536 = 3\,072$) — propriété des suites géométriques de raison 2 ✔.

**Réponse.** Le programme renvoie 1 536 (le dernier terme) · l'erreur est `S = 0` DANS la boucle · corrigé, il renvoie $S = 3 \times \frac{1 - 2^{10}}{1 - 2} = 3\,069$.

**Rappel de cours.** Une initialisation d'accumulateur se fait toujours AVANT la boucle. Dans la boucle, elle annule tout le cumul précédent.

**Erreur fréquente.** Corriger en déplaçant `return` : ce n'est pas l'erreur ici. Le `return` est bien placé, c'est l'initialisation qui est fautive.

## Correction de l'exercice 19 – Type bac : aire sous une courbe par Monte-Carlo

Le domaine est contenu dans le rectangle $[0\,;2] \times [0\,;4]$, dont l'aire vaut $2 \times 4 = 8$.

On tire donc $x$ dans $[0\,;2]$ et $y$ dans $[0\,;4]$, et l'on compte les points vérifiant $y \leq x^2$.

```
from random import random
def aire(N):
    c = 0
for k in range(N):
    x = 2 * random()
    y = 4 * random()
    if y <= x**2:
        c = c + 1
return 8 * c / N
```

Le facteur 8 est essentiel : la fréquence $\frac{c}{N}$ est une PROPORTION du rectangle, qu'il faut multiplier par son aire.

Valeur attendue : l'aire sous $x^2$ entre 0 et 2 vaut $\frac{2^3}{3} = \frac{8}{3} \approx 2{,}667$ (on le verra avec le calcul intégral).

La proportion de points favorables doit donc valoir $\frac{8/3}{8} = \frac{1}{3}$ : environ un point sur trois tombe sous la courbe.

Amélioration de la précision : augmenter $N$. L'erreur décroît comme $\frac{1}{\sqrt{N}}$, donc gagner un facteur 10 de précision demande 100 fois plus de tirages.

Autre amélioration, plus astucieuse : réduire le rectangle. En majorant $f$ par 4 on ne « perd » que les deux tiers des points ; un encadrement plus serré rendrait la simulation plus efficace.

Contrôle de cohérence : l'estimation doit être proche de 2,67 et toujours comprise entre 0 et 8 ✔.

**Réponse.** Rectangle $[0\,;2] \times [0\,;4]$, d'aire 8 · compter les points vérifiant $y \leq x^2$ et

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).

renvoyer $8 \times \frac{c}{N}$ · valeur attendue $\frac{8}{3} \approx 2{,}667$ · précision en $\frac{1}{\sqrt{N}}$.

**Rappel de cours.** Monte-Carlo pour une aire :
aire ≈ aire du rectangle × $\dfrac{\text{points sous la courbe}}{N}$.

**Erreur fréquente.** Oublier de multiplier par l'aire du rectangle : on renverrait 0,333 au lieu de 2,667.

## Correction de l'exercice 20 – Type bac : cumul, seuil et liste

```
def productions(n):
    L = []
    p = 1200
    for k in range(n):
        L.append(p)
        p = 0.98 * p
    return L
def mois(s):
    p = 1200
    S = 0
    n = 0
    while S <= s:
        S = S + p
        p = 0.98 * p
        n = n + 1
    return n
```

Calcul par la formule : $S_n = 1\,200 \times \dfrac{1 - 0{,}98^{\,n}}{1 - 0{,}98} = 60\,000\left(1 - 0{,}98^{\,n}\right)$.

Condition $S_n > 20\,000$ : $1 - 0{,}98^{\,n} > \frac{1}{3}$, soit $0{,}98^{\,n} < 0{,}666\,7$.

Logarithme : $n > \dfrac{\log 0{,}666\,7}{\log 0{,}98} \approx \dfrac{-0{,}176\,1}{-0{,}008\,774} \approx 20{,}07$.

Le seuil est donc dépassé au 21e mois. Contrôle :
$S_{20} = 60\,000(1 - 0{,}98^{20}) \approx 60\,000 \times 0{,}332\,4 \approx 19\,944$ kWh (insuffisant) et $S_{21} \approx 20\,745$ kWh ✔.

Cumul maximal : quand $n$ devient très grand, $0{,}98^{\,n}$ tend vers 0, donc $S_n$ tend vers 60 000 kWh.

L'installation ne produira donc JAMAIS plus de 60 000 kWh au total, même en la laissant fonctionner indéfiniment. C'est la limite d'une somme géométrique de raison inférieure à 1.

Conséquence pour le calcul de rentabilité : un objectif de 70 000 kWh cumulés serait inatteignable avec ce modèle, et la boucle `while` du programme tournerait indéfiniment — d'où l'utilité d'une borne de sécurité.

**Réponse.** Liste par `append` avec `p = 0.98*p` · boucle `while S <= s` · seuil des 20 000 kWh dépassé au 21e mois ($n > 20{,}07$) · cumul plafonné à 60 000 kWh.

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.** Somme géométrique de raison $0 < q < 1$ : le cumul est BORNÉ par $\frac{u_1}{1-q}$, ici $\frac{1\,200}{0{,}02} = 60\,000$.

**Erreur fréquente.** Lancer la boucle `while` avec un seuil supérieur à 60 000 : elle ne s'arrêterait jamais. Tout programme à seuil mérite une borne de sécurité.

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).