

Algorithmique et programmation Python

Corrigé des exercices - Terminale

Corrigé détaillé des exercices du chapitre. Vérifie tes réponses étape par étape et corrige-toi.

1 Lire une boucle for ★★★

- a. **range(1, 6)** exclut 6 : k prend les valeurs 1, 2, 3, 4, 5.
- b. s vaut successivement 1, 5, 14, 30, 55 : le programme affiche **55** ($1 + 4 + 9 + 16 + 25 = 55$).
- c. On remplace $k * k$ par $k ** 3$:

```
s = 0
for k in range(1, 6):
    s = s + k ** 3
print(s)
```

Il affiche 225 ($1 + 8 + 27 + 64 + 125 = 225$).

2 Définir une fonction ★★★

Programme :

```
def f(x):
    return 3 * x ** 2 - 2 * x + 1

print(f(4), f(-1), f(0.5))
```

Sortie obtenue :

```
41 6 0.75
```

À la main : $f(4) = 3 \times 16 - 8 + 1 = 41$; $f(-1) = 3 + 2 + 1 = 6$; $f(0,5) = 0,75 - 1 + 1 = 0,75$. Les résultats concordent.

return renvoie la valeur à l'endroit où la fonction est appelée (on peut la réutiliser dans un calcul) ; **print** affiche seulement à l'écran et la fonction ne renvoie alors rien

d'exploitable.

3 Manipuler une liste ★★★

Les éléments sont 3, 8, 1, 9, 4, 7. La liste a 6 éléments, de somme 32, de maximum 9 ; **L[2]** est l'élément d'indice 2, c'est-à-dire le *troisième* : 1 ; **L[-1]** est le dernier : 7. La deuxième ligne est la moyenne $32 / 6 \approx 5,33$. La troisième garde les éléments strictement supérieurs à 3 (8, 9, 4, 7) et les double : 16, 18, 8, 14.

```
6 32 9 1 7
5.333333333333333
[16, 18, 8, 14]
```

Les indices commencent à 0 : L[0] est le premier élément, L[1] le deuxième, L[2] le troisième.

4 Calculer le terme d'une suite ★★★

- a. $u_1 = 3 \times 2 - 1 = 5$; $u_2 = 3 \times 5 - 1 = 14$; $u_3 = 3 \times 14 - 1 = 41$.
- b. Fonction (la boucle fait n passages) :

```
def terme(n):
    u = 2
    for i in range(n):
        u = 3 * u - 1
    return u

print([terme(n) for n in range(6)])
```

Sortie obtenue :

```
[2, 5, 14, 41, 122, 365]
```

- c. terme(5) renvoie 365. Vérification : la suite $(u_n - 0,5)$ est géométrique de raison 3 et de premier terme 1,5, donc $u_n = 0,5 + 1,5 \times 3^n$ et $u_5 = 0,5 + 1,5 \times 243 = 365$.

5 Un premier seuil ★★★

- a. Baisse de 20 % : coefficient $1 - 0,2 = 0,8$.
- b. Le programme continue tant que la valeur est encore supérieure ou égale à 50 :

```

n = 0
v = 1000
while v >= 50:
    v = 0.8 * v
    n = n + 1
print(n, v)

```

Sortie obtenue :

```
14 43.980465111040026
```

Il s'écoule **14 ans** : la valeur est alors d'environ 43,98 €. Contrôle : $1\,000 \times 0,8^{14} \approx 43,98 < 50$ alors que $1\,000 \times 0,8^{13} \approx 54,98 \geq 50$.

6 Range et indices ★★★

On rappelle que la borne de fin est exclue.

- `list(range(5)) = [0, 1, 2, 3, 4]`
- `list(range(2, 8)) = [2, 3, 4, 5, 6, 7]`
- `list(range(1, 10, 3)) = [1, 4, 7]` (pas de 3)
- `list(range(5, 0, -1)) = [5, 4, 3, 2, 1]` (pas de -1 , 0 exclu)

7 Simuler des lancers de pièce ★★★

- Il simule 20 lancers indépendants d'une pièce équilibrée (1 pour face, 0 pour pile) et compte les faces ; le nombre de faces suit la loi binomiale $B(20 ; 0,5)$.
- Le hasard fait fluctuer le résultat : l'espérance vaut $20 \times 0,5 = 10$, mais le nombre observé s'écarte de 10 en général. Avec la graine choisie, voici la sortie :

```
6 0.3
```

- La graine fixe la suite de nombres pseudo-aléatoires : le programme redonne le même résultat à chaque exécution, ce qui permet de le tester.

8 Suite convergente et précision ★★★

- $\ell = 0,5\ell + 3$ donne $0,5\ell = 3$, donc $\ell = 6$. Pour tout n , $u_{n+1} - 6 = 0,5u_n - 3 = 0,5 \times (u_n - 6)$: la suite $(u_n - 6)$ est géométrique de raison 0,5 et de premier terme 14.

```
b. u = 20
```

```

n = 0
while u - 6 >= 0.01:
    u = 0.5 * u + 3
    n = n + 1
print(n, u)

```

Sortie obtenue :

```
11 6.0068359375
```

Le programme affiche $n = 11$.

c. $u_n - 6 = 14 \times 0,5^n < 0,01$ équivaut à $0,5^n < 1/1400$, soit $n > \ln(1400) / \ln(2) \approx 10,45$. Le plus petit entier est $n = 11$, comme dans le programme.

9 Doubler un capital ★★★

Chaque année, le capital est multiplié par 1,03 :

```

capital = 5000
annees = 0
while capital < 10000:
    capital = capital * 1.03
    annees = annees + 1
print(annees, capital)

```

Sortie obtenue :

```
24 10163.970532302023
```

Il faut **24 ans** (capital $\approx 10163,97$ €). Par le calcul : $5\,000 \times 1,03^n \geq 10\,000$ équivaut à $1,03^n \geq 2$, donc $n \geq \ln 2 / \ln 1,03 \approx 23,45$, soit $n = 24$.

10 Somme des inverses des carrés ★★★

```

s = 0
n = 0
while s <= 1.6:
    n = n + 1
    s = s + 1 / n ** 2

```

```
print(n, s)
```

Sortie obtenue :

```
22 1.6004969333116477
```

Le plus petit n est **22**, avec $S_{22} \approx 1,6005$. La suite (S_n) est croissante (on ajoute un terme positif) et converge vers environ $1,645 > 1,6$: le seuil $1,6$ est donc franchi et la boucle s'arrête. Un seuil de $1,7$, supérieur à la limite, aurait produit une boucle infinie.

11 Dichotomie à la main ★★★

- a. f est continue et strictement croissante sur $[2 ; 3]$ ($f'(x) = 2x > 0$), avec $f(2) = -1 < 0$ et $f(3) = 4 > 0$: d'après le théorème des valeurs intermédiaires (cas strictement monotone), l'équation $f(x) = 0$ a une unique solution, qui est $\sqrt{5}$.
- b. On garde chaque fois la moitié où f change de signe :

Étape	a	b	m	f(m)
1	2,000	3,000	2,500	1,2500
2	2,000	2,500	2,250	0,0625
3	2,000	2,250	2,125	-0,4844

- c. Après la 3^e étape, $\sqrt{5} \in [2,125 ; 2,250]$, amplitude $0,125$. (Valeur exacte $\sqrt{5} \approx 2,2361$, cohérent.)
- d. Après k étapes, l'amplitude vaut $1 / 2^k$. $1 / 2^{13} \approx 0,000122 \geq 10^{-4}$ et $1 / 2^{14} \approx 0,000061 < 10^{-4}$: il faut **14 étapes**.

12 Quatre rectangles à la main ★★★

- a. $h = (1 - 0) / 4 = 0,25$.
- b. Les abscisses sont $0 ; 0,25 ; 0,5 ; 0,75 ; 1$. La fonction x^2 est croissante, donc sous la courbe on prend l'extrémité gauche : $0,25 \times (0 + 0,0625 + 0,25 + 0,5625) = 0,25 \times 0,875 = 0,21875$. Au-dessus, l'extrémité droite : $0,25 \times (0,0625 + 0,25 + 0,5625 + 1) = 0,25 \times 1,875 = 0,46875$.
- c. $0,21875 \leq l \leq 0,46875$, et $0,21875 \leq 1/3 \approx 0,3333 \leq 0,46875$.
- d. Écart : $0,46875 - 0,21875 = 0,25 = h \times (f(1) - f(0))$. Pour avoir $h \times 1 < 0,001$, il faut $1 / n < 0,001$, donc $n > 1000$, soit $n = 1001$ au minimum ($n = 1000$ donne un écart égal à $0,001$, donc pas strictement inférieur).

13 Compter des couples ★★★

```
nb = 0
```

```
for a in range(1, 7):
```

```

for b in range(1, 7):
    if a + b == 7:
        nb = nb + 1
print(nb, nb / 36)

```

Sortie obtenue :

```
6 0.16666666666666666
```

Il y a 6 couples sur 36 : (1 ; 6), (2 ; 5), (3 ; 4), (4 ; 3), (5 ; 2), (6 ; 1). Raisonnement direct : le dé rouge donne n'importe quelle valeur a (6 choix) et il existe alors exactement un b = 7 – a entre 1 et 6 ; donc 6 cas favorables sur 36 et $P = 1/6 \approx 0,1667$.

14 Fréquence et probabilité ★★★

- La fréquence de l'événement « la somme des deux dés vaut 7 » sur 5 000 expériences.
- La sortie est

```
0.1654
```

à comparer à $1/6 \approx 0,1667$: les deux valeurs sont proches, la différence est due aux fluctuations.

- L'écart type de la fréquence vaut $\sqrt{(p(1 - p) / N)} = \sqrt{((1/6) \times (5/6) / 5000)} \approx 0,0053$. On s'attend donc à un écart de l'ordre de quelques millièmes (avec une probabilité d'environ 0,95, de moins de $2 \times 0,0053$).

15 Newton pour une racine carrée ★★★

- $f'(x) = 2x$, donc $x_{n+1} = x_n - (x_n^2 - 7) / (2x_n) = (2x_n^2 - x_n^2 + 7) / (2x_n) = (x_n + 7 / x_n) / 2$.
- $x_1 = (3 + 7/3) / 2 = \mathbf{8/3} = 2,666\dots$; $x_2 = (\mathbf{8/3} + 21/8) / 2 = \mathbf{127/48} \approx 2,6458$.

```

c. x = 3
for i in range(1, 4):
    x = (x + 7 / x) / 2
    print(i, x)
print(7 ** 0.5)

```

Sortie obtenue :

```
1 2.666666666666667
2 2.645833333333333
3 2.6457513123359577
2.6457513110645907
```

x_3 est déjà égal à $\sqrt{7}$ à 10^{-6} près : la convergence est très rapide.

16 Newton sur une équation du troisième degré ★★★

- a. f est dérivable sur $\mathbb{R}$ avec $f'(x) = 3x^2 + 2 > 0$: f est strictement croissante, continue, de limites $-\infty$ et $+\infty$; elle s'annule donc exactement une fois. Comme $f(1) = -2 < 0$ et $f(2) = 7 > 0$, $\alpha \in [1 ; 2]$.
- b. La suite de Newton est $x_{n+1} = x_n - f(x_n) / f'(x_n)$; f' ne s'annule jamais, aucune division par zéro.

```
def f(x):
    return x ** 3 + 2 * x - 5

def fp(x):
    return 3 * x ** 2 + 2

x = 1.5
n = 0
while abs(f(x)) > 1e-12:
    x = x - f(x) / fp(x)
    n = n + 1
print(n, x)
```

Sortie obtenue :

```
4 1.3282688556686084
```

- c. Le programme s'arrête au bout de 4 itérations avec $\alpha \approx 1,328269$. Vérification : $f(\alpha) \approx 0$.

17 Dichotomie et exponentielle ★★★

- a. f est continue et dérivable sur $[0 ; 2]$, $f'(x) = e^x + 1 > 0$: f est strictement croissante. $f(0) = 1 + 0 - 3 = -2 < 0$ et $f(2) = e^2 - 1 \approx 6,39 > 0$. Le théorème des valeurs intermédiaires (fonction strictement monotone) donne une unique solution $\alpha \in [0 ; 2]$.

```

b. from math import exp

def f(x):
    return exp(x) + x - 3

a, b = 0, 2
etapes = 0
while b - a > 1e-6:
    m = (a + b) / 2
    if f(a) * f(m) <= 0:
        b = m
    else:
        a = m
    etapes = etapes + 1
print(a, b, etapes)

```

Sortie obtenue :

```
0.7920598983764648 0.7920608520507812 21
```

Donc $\alpha \approx 0,792060$.

- c. $(b - a) / e = 2 / 10^{-6} = 2 \times 10^6$. Comme $2^{20} = 1\,048\,576 < 2 \times 10^6 < 2^{21} = 2\,097\,152$, il faut **21 étapes**, ce que le programme confirme (dernier nombre : 21).

18 Estimer $\ln 2$ par les rectangles ★★★

- Avec $h = 1/n$ et $x_k = 1 + k/n$, la fonction étant *décroissante*, le rectangle de hauteur $f(x_{k+1})$ est sous la courbe : le minorant est $h \times [f(x_1) + \dots + f(x_n)]$ et le majorant $h \times [f(x_0) + \dots + f(x_{n-1})]$.
- La différence est $h \times (f(x_0) - f(x_n))$ (les autres termes se simplifient) $= (1/n) \times (1 - 1/2) = 1 / (2n)$.
- $1 / (2n) < 0,001$ équivaut à $n > 500$, soit $n = 501$ au minimum. Avec $n = 500$:

```

def rect_dec(f, a, b, n):
    h = (b - a) / n
    haut = sum(f(a + k * h) for k in range(n))
    bas = sum(f(a + (k + 1) * h) for k in range(n))
    return h * bas, h * haut

```

```
print(rect_dec(lambda x: 1 / x, 1, 2, 500))
```

Sortie obtenue :

```
(0.6926474305598204, 0.6936474305598204)
```

Le programme donne un encadrement de $\ln 2$: (0.6926474305598204, 0.6936474305598204) ($\ln 2 \approx 0,693147$), d'écart 0,001.

19 Seuil de probabilité ★★★

- a. L'événement contraire est « aucun succès » de probabilité $0,7^n$: la probabilité cherchée est $1 - 0,7^n$.

```
b. n = 0
   p = 1
   while 1 - p < 0.99:
       n = n + 1
       p = p * 0.7
   print(n, 1 - p)
```

Sortie obtenue :

```
13 0.9903110989593
```

Il faut **n = 13** tirs.

- c. $1 - 0,7^n \geq 0,99$ équivaut à $0,7^n \leq 0,01$. Comme $\ln 0,7 < 0$, on obtient $n \geq \ln(0,01) / \ln(0,7) \approx 12,91$, donc $n = 13$. (Contrôle : $0,7^{12} \approx 0,0138 > 0,01$ et $0,7^{13} \approx 0,0097$.)

20 Estimer π par Monte-Carlo ★★★

- a. L'aire du carré vaut 1 et celle du quart de disque $\pi \times 1^2 / 4$: la probabilité est $\pi / 4$.
b. Le programme compte les points tels que $x^2 + y^2 \leq 1$ (dans le quart de disque) ; la fréquence tend vers $\pi / 4$, donc $4 \times \text{fréquence} \approx \pi$. Sortie :

```
3.1348
```

- c. Pour la fréquence, l'écart type est $\sqrt{p(1 - p) / N}$ avec $p = \pi / 4 \approx 0,785$: environ 0,0013, ce qui donne pour $4 \times \text{fréquence}$ une précision de l'ordre de 0,0052 (deux décimales justes environ). Pour gagner une décimale, il faut diviser l'écart par 10, donc multiplier N par 100 : 10 millions de points.

21 Arrangements et combinaisons ★★★

- a. Première ligne : on forme des mots de 4 lettres *distinctes* avec A, B, C, D, E (l'ordre compte) : $5 \times 4 \times 3 \times 2 = 120$. Deuxième ligne : mains de 5 cartes dans un jeu de 32, l'ordre ne comptant pas : $C(32, 5) = 32 \times 31 \times 30 \times 29 \times 28 / 120 = 201\,376$. Troisième ligne : $C(5, 3) = 10$ (le programme le recalcule en listant les 10 sous-ensembles de 3 lettres).
- b. Sortie obtenue :

```
120
201376
10 10
```

Lister les 201 376 mains consommerait de la mémoire et du temps pour rien : on n'a besoin que du *nombre*, que **comb** donne immédiatement.

22 Loi binomiale par programme ★★★

- a. $P(X = k) = C(n, k) \times p^k \times (1 - p)^{n-k}$.
- b. Programme :

```
from math import comb

def binom(n, p, k):
    return comb(n, k) * p ** k * (1 - p) ** (n - k)

print(binom(10, 0.3, 3))
print(sum(binom(10, 0.3, k) for k in range(3)))
print(1 - sum(binom(10, 0.3, k) for k in range(2)))
```

Sortie obtenue :

```
0.2668279319999998
0.3827827863999998
0.8506916541
```

On lit $P(X = 3) \approx 0,2668$; $P(X \leq 2) \approx 0,3828$ (somme de $k = 0, 1, 2$) ; $P(X \geq 2) = 1 - P(X \leq 1) \approx 0,8507$.

- c. $E(X) = 10 \times 0,3 = 3$. En programmant **sum(k * binom(10, 0.3, k) for k in range(11))**, on obtient 3,0000 : la valeur attendue.

23 Problème : un parc de vélos ★★★

- a. Retirer 10 % revient à multiplier par 0,9, puis on ajoute 80 : $u_{n+1} = 0,9u_n + 80$.
- b. $v_{n+1} = 0,9u_n + 80 - 800 = 0,9u_n - 720 = 0,9 \times (u_n - 800) = 0,9v_n$. La suite (v_n) est géométrique de raison 0,9 et de premier terme $v_0 = 200$. Donc $u_n = 800 + 200 \times 0,9^n$. Comme $0 < 0,9 < 1$, $0,9^n$ tend vers 0 et la limite est 800.

```
c. u = 1000
   n = 0
   while u >= 810:
       u = 0.9 * u + 80
       n = n + 1
   print(n, u)
```

Sortie obtenue :

```
29 809.4202573944928
```

Il faut $n = 29$: c'est en 2054 qu'il y aura moins de 810 vélos ($u_{29} \approx 809,42$).
Contrôle : $200 \times 0,9^n < 10$ équivaut à $0,9^n < 0,05$, soit $n > 28,43$.