Plaquette 4 – Simulations aléatoires et statistiques

Algorithmique et Python — Seconde

Corrections détaillées

Boîte à outils

Les briques du hasard.

Instruction	Effet
from random import random	importe le générateur décimal
from random import randint	importe le générateur entier
random()	un décimal au hasard dans $[0\,;1[$
randint(a, b)	un entier au hasard de a à b **inclus**
random() < p	vrai avec la probabilité p
choice(L)	un élément au hasard de la liste L

Simuler une expérience de probabilité p. On écrit *if random() < p:* — la condition est vraie exactement avec la probabilité p, puisque *random()* est uniforme sur $[0\,;1[$.

Structure d'une estimation de probabilité.

```
def estime(n):
    succes = 0
    for i in range(n):
        # une répétition de l'expérience
        if condition:
            succes = succes + 1
    return succes / n
```

La valeur renvoyée est une **fréquence**, à diviser par n — et non un effectif.

Précision d'une simulation. L'écart typique entre la fréquence obtenue et la probabilité réelle est de l'ordre de $\frac{1}{\sqrt{n}}$: il faut multiplier n par 100 pour gagner un chiffre.

n	Précision typique
100	±0,1
10 000	±0,01
1 000 000	±0,001

Indicateurs sur une liste simulée.

Indicateur	Calcul
moyenne	somme divisée par l'effectif
étendue	*max(L) - min(L)*
fréquence d'une valeur	*L.count(v) / len(L)*

Deux réflexes de contrôle.

— Comparer le résultat simulé à la valeur **théorique** quand on la connaît.

— Relancer le programme : un résultat qui ne **varie pas** du tout révèle un bug.

Correction de l’exercice 1 – Simuler un dé

— **Simulation d’un lancer** : *randint(1, 6)* ✓

```
from random import randint

def frequenceSix(n):
    c = 0
    for i in range(n):
        if randint(1, 6) == 6:
            c = c + 1
    return c / n
```

— **Probabilité théorique** : $\frac{1}{6} \approx 0{,}1667$ ✓

— **Effectif attendu pour** $n = 6000$: $\frac{6000}{6} = 1000$ six ✓

— **Fréquence attendue** :

$$\approx 0{,}167$$

— **Fluctuation typique** : $\frac{1}{\sqrt{6000}} \approx 0{,}0129$ ✓

— **Fourchette plausible** : environ $[0{,}154\,;0{,}180]$ ✓

— **En effectifs** : entre 924 et 1080 six ✓

— **Piège de l’intervalle** : *randint(1, 5)* exclurait le 6, la fréquence serait 0 ✓

— **Autre piège** : *randint(0, 6)* donnerait 7 issues, et une fréquence de $\frac{1}{7} \approx 0{,}143$ ✓

— **Piège du* return** *: renvoyer* c *au lieu de* c / n* donnerait 1000, un effectif ✓

— **Deux exécutions consécutives** : elles donnent des résultats **différents** ✓

— **Explication** : chaque appel à *randint* produit un nouveau tirage ✓

— **Vérification de l’équilibre** : on peut compter les six faces et vérifier qu’elles sont proches de 1000 chacune ✓

— **Somme de contrôle** : les six effectifs doivent totaliser exactement 6000 ✓

— **Pour estimer la face la plus fréquente** : construire une liste de six compteurs ✓

Réponse. La fréquence obtenue doit être proche de $\frac{1}{6} \approx 0{,}167$, soit environ 1000 six sur 6000 lancers.

Correction de l’exercice 2 – Somme de deux dés

— **Simulation** : deux appels à *randint* ✓

```
from random import randint

def estimeSept(n):
    c = 0
    for i in range(n):
        if randint(1, 6) + randint(1, 6) == 7:
            c = c + 1
    return c / n
```

— **Piège majeur** : simuler **un** dé et le doubler ✓

— **Conséquence de ce piège** : la somme serait toujours **paire**, donc jamais 7 ✓

— **Nécessité** : deux appels **distincts** à *randint* ✓
— **Dénombrement théorique** : 36 couples équiprobables ✓
— **Couples de somme** 7 : (1 ; 6), (2 ; 5), (3 ; 4), (4 ; 3), (5 ; 2), (6 ; 1) ✓
— **Comptage** : 6 couples ✓
— **Probabilité théorique** :

$$\frac{6}{36} = \frac{1}{6} \approx 0{,}1667$$

— **Fréquence attendue pour** $n = 10\,000$: environ 0,167 ✓
— **Fluctuation typique** : $\frac{1}{\sqrt{10\,000}} = 0{,}01$ ✓
— **Fourchette plausible** : [0,157 ; 0,177] ✓
— **Pour la somme** 2 : un seul couple, donc $\frac{1}{36} \approx 0{,}0278$ ✓
— **Pour la somme** 12 : également $\frac{1}{36}$ ✓
— **Somme la plus probable** : 7 ✓
— **Contrôle de la loi complète** : les effectifs sont 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 5 ; 4 ; 3 ; 2 ; 1 ✓
— **Somme de contrôle** : 36 ✓
— **Adaptation pour trois dés** : ajouter un troisième appel, avec 216 issues ✓

Réponse. La probabilité théorique est $\frac{6}{36} = \frac{1}{6} \approx 0{,}167$.

Correction de l’exercice 3 – Tirage dans une urne

— **Modélisation d’un tirage** : *random() < 0.7* est vrai avec la probabilité 0,7 ✓

```
from random import random

def deuxRouges(n):
    c = 0
    for i in range(n):
        if random() < 0.7 and random() < 0.7:
            c = c + 1
    return c / n
```

— **Rôle du* and*** : les deux tirages doivent réussir ✓
— **Avec remise** : la probabilité est la **même** aux deux tirages ✓
— **Probabilité théorique** :

$$0{,}7 \times 0{,}7 = 0{,}49$$

— **Fréquence attendue pour** $n = 10\,000$: environ 0,49 ✓
— **Effectif attendu** : 4900 ✓
— **Fluctuation typique** : 0,01 ✓
— **Fourchette plausible** : [0,48 ; 0,50] ✓
— **Piège du* and *mal écrit :* if random() < 0.7 and 0.7 *est syntaxiquement valide mais toujours** vrai ✓
— **Explication** : 0,7 est considéré comme vrai, étant non nul ✓
— **Simulation alternative avec une liste** : *urne = ["R"] 7 + ["V"] 3*, puis *choice(urne)* ✓
— **Avantage de cette version** : elle décrit fidèlement l’urne ✓
— **Pour un tirage sans remise** : il faudrait **retirer** la boule de la liste ✓
— **Probabilité théorique sans remise** : $\frac{7}{10} \times \frac{6}{9} = \frac{42}{90} \approx 0{,}4667$ ✓
— **Comparaison** : 0,4667 < 0,49 ✓

— **Interprétation** : sans remise, deux rouges sont **moins** probables ✓
— **Écart** : environ 2,3 points, détectable avec $n = 10\,000$ ✓

Réponse. $0{,}7^2 = 0{,}49$ avec remise (contre $\frac{42}{90} \approx 0{,}467$ sans remise).

Correction de l'exercice 4 – Estimer π par Monte-Carlo

— **Principe** : comparer l'aire du quart de disque à celle du carré ✓

```
from random import random
```

```
def estimePi(n):
    dedans = 0
    for i in range(n):
        x = random()
        y = random()
        if x * x + y * y < 1:
            dedans = dedans + 1
    return 4 * dedans / n
```

— **Domaine des points** : le carré $[0\,;1[^2$ ✓
— **Condition d'appartenance au quart de disque** : $x^2 + y^2 < 1$ ✓
— **Aire du carré** : 1 ✓
— **Aire du quart de disque** : $\frac{\pi \times 1^2}{4} = \frac{\pi}{4}$ ✓
— **Probabilité de tomber dedans** : $\frac{\pi}{4} \approx 0{,}7854$ ✓
— **D'où la formule** : $\pi \approx 4 \times \frac{\text{dedans}}{n}$ ✓
— **Estimation attendue pour** $n = 10\,000$:

$$\approx 3{,}14$$

— **Effectif attendu** : environ 7854 points dedans ✓
— **Fluctuation sur la fréquence** : $\approx 0{,}01$ ✓
— **Fluctuation sur** π : $4 \times 0{,}01 = 0{,}04$ ✓
— **Fourchette plausible** : $[3{,}10\,;3{,}18]$ ✓
— **Piège du facteur** 4 : l'oublier donnerait $\approx 0{,}785$ ✓
— **Piège de la condition** : écrire $x + y < 1$ délimiterait un **triangle**, de probabilité 0,5 ✓
— **Résultat dans ce cas** : ≈ 2, très loin de π ✓
— **Nombre de points pour trois décimales** : il faudrait $n \approx 16$ millions ✓
— **Contrôle** : $4 \times \frac{1}{\sqrt{16\,000\,000}} = \frac{4}{4000} = 0{,}001$ ✓
— **Verdict sur la méthode** : élégante mais **très lente** ✓

Réponse. $\pi \approx 4 \times \frac{\text{points dedans}}{n}$, car la probabilité de tomber dans le quart de disque vaut $\frac{\pi}{4}$.

Correction de l'exercice 5 – Marche aléatoire

— **Simulation d'un pas** : $+1$ ou -1 selon un tirage ✓

```
from random import random
```

```
def marche(n):
    x = 0
    for i in range(n):
```

```
        if random() < 0.5:
            x = x + 1
        else:
            x = x - 1
    return x
```

— **Position moyenne attendue** :

$$0$$

— **Justification** : les deux directions sont **symétriques** ✓
— **Attention** : cela ne signifie **pas** qu'on finit souvent en 0 ✓
— **Parité de la position finale** : après 100 pas, elle est **paire** ✓
— **Justification** : 100 pas donnent une somme de 100 termes impairs ✓
— **Positions possibles** : de -100 à 100, de deux en deux ✓
— **Éloignement typique** : de l'ordre de $\sqrt{100} = 10$ ✓
— **Fourchette plausible** : environ $[-20\,;20]$ ✓
— **Probabilité de finir exactement en** 0 : environ 8 % ✓
— **Probabilité de finir en** 100 : $\left(\frac{1}{2}\right)^{100}$, soit une chance sur 10^{30} ✓
— **Conclusion** : l'extrême est inatteignable en pratique ✓
— **Loi de l'éloignement** : la distance typique croît comme $\sqrt{n}$, non comme n ✓
— **Contrôle** : après 10 000 pas, l'éloignement typique serait de 100 ✓
— **Moyenne sur de nombreuses simulations** : elle tend vers 0 ✓
— **Moyenne des valeurs absolues** : elle tend vers $\approx 0{,}8\sqrt{n} \approx 8$ pour $n = 100$ ✓
— **Application** : ce modèle décrit le mouvement brownien et les cours de bourse ✓

Réponse. La position moyenne attendue est 0, mais l'éloignement typique est de l'ordre de $\sqrt{100} = 10$.

Correction de l'exercice 6 – Pile ou face : plus longue série

— **Structure** : deux compteurs, l'actuel et le record ✓

```
from random import random
```

```
def plusLongueSerie(n):
    record = 0
    courant = 0
    for i in range(n):
        if random() < 0.5:
            courant = courant + 1
            if courant > record:
                record = courant
        else:
            courant = 0
    return record
```

— *Rôle de* courant*** : la longueur de la série en cours ✓
— *Rôle de* record*** : le maximum vu jusqu'ici ✓
— **Remise à zéro** : dès qu'un face apparaît ✓
— **Mise à jour du record** : à chaque allongement de la série courante ✓
— **Valeur typique pour** $n = 50$:

environ 5 ou 6

— **Justification** : la plus longue série croît comme $\log_2 n$ ✓

— **Contrôle** : $\log_2 50 \approx 5{,}64$ ✓
— **Probabilité d'une série de** 10 : très faible, environ 2,4 % ✓
— **Probabilité d'aucune série de** 3 : faible aussi, les séries courtes étant fréquentes ✓
— **Piège de la remise à zéro** : l'oublier donnerait le **nombre total** de piles ✓
— **Valeur dans ce cas** : environ 25 ✓
— **Piège de la mise à jour** : comparer seulement à la fin perdrait les records intermédiaires ✓
— **Exemple** : avec *PPPFFPP*, le record est 3, mais *courant* finit à 2 ✓
— **Pour** $n = 200$: le record typique monterait à 7 ou 8 ✓
— **Contrôle** : $\log_2 200 \approx 7{,}64$ ✓
— **Application** : ce calcul sert à détecter des séquences **inventées**, où les séries sont trop courtes ✓

Réponse. Environ 5 ou 6 pour 50 lancers, la plus longue série croissant comme $\log_2 n$.

Correction de l'exercice 7 – Indicateurs d'une simulation

— **Structure** : construire une liste, puis l'analyser ✓

```
from random import randint

L = []
for i in range(1000):
    L.append(randint(1, 6) + randint(1, 6))

s = 0
for x in L:
    s = s + x
print(s / len(L))
print(max(L) - min(L))
```

— **Moyenne théorique d'un dé** : $\dfrac{1+2+3+4+5+6}{6} = 3{,}5$ ✓
— **Moyenne théorique de la somme** :

$$2 \times 3{,}5 = 7$$

— **Fluctuation attendue sur la moyenne** : de l'ordre de $\dfrac{2{,}42}{\sqrt{1000}} \approx 0{,}077$ ✓
— **Fourchette plausible** : environ $[6{,}85\,;7{,}15]$ ✓
— **Étendue théorique maximale** : $12 - 2 = 10$ ✓
— **Étendue attendue avec** 1000 **lancers** :

$$10$$

— **Justification** : les sommes 2 et 12 ont chacune une probabilité de $\dfrac{1}{36}$ ✓
— **Nombre attendu de chaque extrême** : ≈ 28 ✓
— **Contrôle** : $\dfrac{1000}{36} \approx 27{,}8$ ✓
— **Conclusion** : les deux extrêmes apparaissent presque sûrement ✓
— **Probabilité de n'obtenir aucun** 12 : $\left(\dfrac{35}{36}\right)^{1000} \approx 8{,}6 \times 10^{-13}$ ✓
— **Avec seulement** 20 **lancers** : l'étendue serait souvent inférieure à 10 ✓
— **Contrôle** : $\left(\dfrac{35}{36}\right)^{20} \approx 0{,}569$ de chances de n'avoir aucun 12 ✓
— **Écart-type théorique de la somme** : $\approx 2{,}415$ ✓

— **Fréquence attendue du** 7 : $\frac{1}{6}$, soit environ 167 lancers ✓

— **Instruction pour la vérifier** : *L.count(7)* ✓

Réponse. Moyenne proche de 7 et étendue égale à 10 (les sommes 2 et 12 apparaissant presque sûrement).

Correction de l'exercice 8 – Probabilité conditionnelle simulée

— **Principe** : ne compter que les tirages où la condition est réalisée ✓

```
from random import randint

def conditionnelle(n):
    pairs = 0
    favorables = 0
    for i in range(n):
        d = randint(1, 6)
        if d % 2 == 0:
            pairs = pairs + 1
            if d > 4:
                favorables = favorables + 1
    return favorables / pairs
```

— **Dénominateur** : le nombre de tirages **pairs**, non le nombre total ✓

— **Résultats pairs** : 2, 4, 6 ✓

— **Parmi eux, ceux supérieurs à** 4 : le seul 6 ✓

— **Probabilité théorique** :

$$\frac{1}{3} \approx 0{,}3333$$

— **Fréquence attendue pour** $n = 9000$: environ 0,333 ✓

— **Nombre de pairs attendu** : 4500 ✓

— **Nombre de favorables attendu** : 1500 ✓

— **Contrôle** : $\frac{1500}{4500} = \frac{1}{3}$ ✓

— **Piège majeur** : diviser par n au lieu de *pairs* ✓

— **Résultat dans ce cas** : $\frac{1500}{9000} = \frac{1}{6}$ ✓

— **Interprétation de cette valeur** : c'est P(pair et > 4), non la probabilité conditionnelle ✓

— **Relation entre les deux** : $\frac{1}{6} = \frac{1}{3} \times \frac{1}{2}$ ✓

— **Formule générale** : $P(A \text{ sachant } B) = \frac{P(A \text{ et } B)}{P(B)}$ ✓

— **Contrôle** : $\frac{1/6}{1/2} = \frac{1}{3}$ ✓

— **Risque si* pairs *vaut* 0* : division par zéro ✓

— **Protection** : tester *if pairs == 0: return None* ✓

Réponse. $\frac{1}{3} \approx 0{,}333$ — il faut bien diviser par le nombre de résultats **pairs**.

Correction de l'exercice 9 – Anniversaires

— **Simulation d'un groupe** : 23 tirages dans 1 à 365 ✓

— **Détection d'une coïncidence** : comparer la longueur de la liste à celle de l'ensemble des valeurs distinctes ✓

```
from random import randint
```

```
def coincidence(n, taille):
    c = 0
    for k in range(n):
        jours = []
        for i in range(taille):
            jours.append(randint(1, 365))
        if len(set(jours)) < taille:
            c = c + 1
    return c / n
```

— *Rôle de* set* **: il supprime les** doublons** ✓
— **Condition de coïncidence** : la taille diminue après suppression ✓
— **Calcul théorique** : on passe par le contraire, « tous différents » ✓
— **Probabilité de tous différents** : $\frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{343}{365}$ ✓
— **Valeur** : $\approx 0{,}4927$ ✓
— **Probabilité cherchée** :

$$\approx 1 - 0{,}4927 = 0{,}5073$$

— **Résultat célèbre** : avec 23 personnes, il y a **plus** d'une chance sur deux ✓
— **Fréquence attendue pour** $n = 10\,000$ **simulations** : environ 0,507 ✓
— **Fluctuation** : $\pm 0{,}01$ ✓
— **Intuition trompeuse** : on attendrait plutôt 183 personnes ✓
— **Explication** : le nombre de **paires** croît comme le carré de l'effectif ✓
— **Nombre de paires pour** 23 **personnes** : $\frac{23 \times 22}{2} = 253$ ✓
— **Estimation rapide** : $\frac{253}{365} \approx 0{,}693$, du bon ordre de grandeur ✓
— **Avec** 50 **personnes** : la probabilité monte à $\approx 0{,}970$ ✓
— **Avec** 10 **personnes** : $\approx 0{,}117$ ✓
— **Seuil des** 99 % : atteint à 57 personnes ✓

Réponse. $\approx 0{,}507$: avec 23 personnes, la coïncidence est plus probable qu'improbable.

Correction de l'exercice 10 – Jeu équitable ?

— **Structure** : accumuler les gains nets ✓

```
from random import randint
```

```
def gainMoyen(n):
    total = 0
    for i in range(n):
        s = randint(1, 6) + randint(1, 6)
        if s > 9:
            total = total + 6
        else:
            total = total - 2
    return total / n
```

— **Gain net en cas de victoire** : $8 - 2 = 6$ € ✓
— **Gain net en cas d'échec** : -2 € ✓
— **Sommes gagnantes** : 10, 11, 12 ✓
— **Effectifs correspondants** : 3, 2, 1 ✓

— **Total** : 6 issues sur 36 ✓
— **Probabilité de gagner** : $\frac{1}{6}$ ✓
— **Gain moyen théorique** : $6 \times \frac{1}{6} - 2 \times \frac{5}{6}$ ✓
— **Calcul** : $1 - \frac{10}{6} = 1 - 1{,}6667$ ✓
— **Résultat** :

$$\approx -0{,}667 \text{ €}$$

— **Conclusion** : le jeu est **défavorable** au joueur ✓
— **Perte moyenne** : environ 0,67 € par partie ✓
— **En proportion de la mise** : 33,3 % ✓
— **Fréquence attendue pour** $n = 10\,000$: un gain moyen proche de $-0{,}67$ € ✓
— **Perte totale attendue** : environ 6670 € ✓
— **Gain rendant le jeu équitable** : il faut g tel que $(g-2) \times \frac{1}{6} = 2 \times \frac{5}{6}$ ✓
— **Résolution** : $g - 2 = 10$, donc $g = 12$ € ✓
— **Contrôle** : $10 \times \frac{1}{6} - 2 \times \frac{5}{6} = \frac{10-10}{6} = 0$ ✓

Réponse. Gain moyen théorique $= 6 \times \frac{1}{6} - 2 \times \frac{5}{6} \approx -0{,}67$ € : le jeu est défavorable (il faudrait 12 € de gain).

Correction de l’exercice 11 – Estimer une aire

— **Principe** : la fréquence de points vérifiant la condition estime l’aire ✓

```
from random import random

def aire(n):
    dedans = 0
    for i in range(n):
        x = random()
        y = random()
        if y < x * x:
            dedans = dedans + 1
    return dedans / n
```

— **Justification** : l’aire du carré valant 1, la proportion **est** l’aire ✓
— **Aire exacte** : c’est l’intégrale de x^2 de 0 à 1 ✓
— **Valeur** :

$$\frac{1}{3} \approx 0{,}3333$$

— **Justification élémentaire** : la primitive de x^2 est $\frac{x^3}{3}$ ✓
— **Fréquence attendue pour** $n = 10\,000$: environ 0,333 ✓
— **Fluctuation** : $\pm 0{,}01$ ✓
— **Effectif attendu** : environ 3333 points ✓
— **Piège de l’inégalité** : écrire $y < x$ donnerait l’aire du triangle, soit 0,5 ✓
— **Autre condition instructive** : $y < x\ 3$ donnerait $\frac{1}{4}$ ✓
— **Généralisation** : $y < x\ k$ donne $\frac{1}{k+1}$ ✓
— **Contrôle pour** $k = 1$: $\frac{1}{2}$ ✓

— **Contrôle pour** $k = 2$: $\frac{1}{3}$ ✓

— **Contrôle pour** $k = 3$: $\frac{1}{4}$ ✓

— **Région complémentaire** : $y > x^2$ a pour aire $\frac{2}{3}$ ✓

— **Contrôle** : $\frac{1}{3} + \frac{2}{3} = 1$ ✓

— **Intérêt de la méthode** : elle fonctionne pour des régions dont on ne sait **pas** calculer l'aire ✓

Réponse. La fréquence obtenue approche $\frac{1}{3} \approx 0{,}333$, l'aire exacte sous la parabole.

Correction de l'exercice 12 – Comparer deux stratégies

— **Structure** : deux simulations indépendantes ✓

```
from random import random

def gain(n, p, montant):
    total = 0
    for i in range(n):
        if random() < p:
            total = total + montant
    return total / n

print(gain(100000, 0.4, 10))
print(gain(100000, 0.7, 5))
```

— **Gain moyen théorique de A** : $0{,}4 \times 10$ ✓

— **Résultat** :

$$4 \text{ €}$$

— **Gain moyen théorique de B** : $0{,}7 \times 5$ ✓

— **Résultat** :

$$3{,}50 \text{ €}$$

— **Conclusion** : la stratégie **A** est meilleure en moyenne ✓

— **Écart** : 0,50 € par partie ✓

— **Écart relatif** : $\frac{0{,}50}{3{,}50} \approx 14{,}3\,\%$ ✓

— **Intuition trompeuse** : B gagne **plus souvent** ✓

— **Mais** : A rapporte le double à chaque victoire ✓

— **Fluctuation pour** $n = 100\,000$: de l'ordre de $\frac{10}{\sqrt{100\,000}} \approx 0{,}032$ € pour A ✓

— **Conclusion sur la simulation** : l'écart de 0,50 € est largement **détectable** ✓

— **Avec seulement** $n = 100$: la fluctuation atteindrait 1 €, l'écart serait noyé ✓

— **Variabilité comparée** : A est plus **risquée**, ses gains valant 0 ou 10 ✓

— **Écart-type de A** : $10\sqrt{0{,}4 \times 0{,}6} \approx 4{,}90$ € ✓

— **Écart-type de B** : $5\sqrt{0{,}7 \times 0{,}3} \approx 2{,}29$ € ✓

— **Conclusion nuancée** : A rapporte plus en moyenne, mais B est plus **régulière** ✓

— **Choix selon le contexte** : sur une seule partie, B est plus sûre ; à long terme, A est meilleure ✓

Réponse. A rapporte $0{,}4 \times 10 = 4$ € en moyenne contre $0{,}7 \times 5 = 3{,}50$ € pour B : A est meilleure, mais plus risquée.

Correction de l’exercice 13 – Fluctuation d’échantillonnage

— **Structure** : deux boucles imbriquées ✓

```
from random import random
```

```
def sorties(repetitions, n):
    c = 0
    for k in range(repetitions):
        piles = 0
        for i in range(n):
            if random() < 0.5:
                piles = piles + 1
        f = piles / n
        if f < 0.4 or f > 0.6:
            c = c + 1
    return c
```

— **Intervalle de fluctuation pour** $n = 100$: $\left[0{,}5 - \frac{1}{10} ; 0{,}5 + \frac{1}{10}\right] = [0{,}4 ; 0{,}6]$ ✓

— **Niveau de confiance associé** : 95 % ✓

— **Proportion de sorties attendue** : 5 % ✓

— **Nombre attendu** :

$$0{,}05 \times 200 = 10 \text{ sorties}$$

— **Fourchette plausible** : entre 4 et 17 environ ✓

— **Contrôle** : la fluctuation sur ce compte est de l’ordre de $\sqrt{200 \times 0{,}05 \times 0{,}95} \approx 3{,}1$ ✓

— **Interprétation du seuil de** 95 % : il **annonce** ces 5 % d’échecs ✓

— **Effectifs limites** : entre 40 et 60 piles sur 100 ✓

— **Probabilité exacte d’être hors de ces bornes** : environ 3,5 % ✓

— **Écart avec les** 5 % **annoncés** : la formule $\frac{1}{\sqrt{n}}$ est **prudente** ✓

— **Nombre attendu avec la valeur exacte** : ≈ 7 sorties ✓

— **Conclusion** : observer entre 4 et 15 sorties est parfaitement normal ✓

— **Résultat qui invaliderait la simulation** : 80 sorties sur 200 ✓

— **Diagnostic dans ce cas** : une erreur de probabilité ou de bornes ✓

— **Avec** $n = 400$: l’intervalle se resserre à $[0{,}45 ; 0{,}55]$ ✓

— **Nombre de sorties attendu** : toujours environ 10, le niveau de confiance étant inchangé ✓

Réponse. Environ 10 sorties sur 200 (soit les 5 % annoncés par le seuil de 95 %).

Correction de l’exercice 14 – Tirage sans remise

— **Simulation** : tirer, puis retirer de la liste ✓

```
from random import choice
```

```
def sommePaire(n):
    c = 0
    for k in range(n):
        urne = [1, 2, 3, 4, 5]
        a = choice(urne)
        urne.remove(a)
        b = choice(urne)
        if (a + b) % 2 == 0:
            c = c + 1
```

```
    return c / n
```

— **Importance de recréer l'urne** : à **chaque** répétition ✓
— **Conséquence de l'oubli** : l'urne se viderait et le programme planterait ✓
— **Rôle de* remove*** : simuler l'absence de remise ✓
— **Dénombrement théorique** : $\frac{5 \times 4}{2} = 10$ paires ✓
— **Somme paire** : deux pairs ou deux impairs ✓
— **Deux pairs parmi** $\{2\,;4\}$: une seule paire ✓
— **Deux impairs parmi** $\{1\,;3\,;5\}$: 3 paires ✓
— **Total** : 4 paires favorables ✓
— **Probabilité théorique** :

$$\frac{4}{10} = 0{,}4$$

— **Fréquence attendue** : environ 0,40 ✓
— **Somme impaire** : 0,6 ✓
— **Contrôle par dénombrement** : $2 \times 3 = 6$ paires mixtes ✓
— **Contrôle** : $4 + 6 = 10$ ✓
— **Piège de l'ordre** : *choice* tire au hasard, l'ordre n'influe pas sur la somme ✓
— **Variante avec remise** : la probabilité passerait à $\frac{9+4}{25} = 0{,}52$ ✓
— **Contrôle** : 3^2 couples d'impairs et 2^2 couples de pairs ✓
— **Écart** : 12 points, bien détectable en simulation ✓

Réponse. $\frac{4}{10} = 0{,}4$ (contre 0,52 avec remise).

Correction de l'exercice 15 – Temps d'attente

— **Structure** : une boucle *while*, le nombre de lancers étant inconnu ✓

```
from random import randint
```

```
def attente():
    n = 0
    resultat = 0
    while resultat != 6:
        resultat = randint(1, 6)
        n = n + 1
    return n
```

— **Initialisation de* resultat*** : à une valeur différente de 6, pour que la boucle démarre ✓
— **Alternative plus propre** : une boucle *while True* avec un *break* ✓
— **Moyenne théorique** :

6 lancers

— **Justification** : pour une probabilité p, l'attente moyenne vaut $\frac{1}{p}$ ✓
— **Contrôle** : $\frac{1}{1/6} = 6$ ✓
— **Valeur la plus fréquente** : 1 lancer, avec une probabilité de $\frac{1}{6}$ ✓
— **Surprise** : la valeur la plus probable n'est **pas** la moyenne ✓
— **Explication** : la distribution est très **asymétrique** ✓

— **Probabilité de réussir au premier lancer** : $\approx 16{,}7\,\%$ ✓
— **Probabilité de dépasser** 10 **lancers** : $\left(\frac{5}{6}\right)^{10} \approx 16{,}2\,\%$ ✓
— **Probabilité de dépasser** 20 **lancers** : $\approx 2{,}6\,\%$ ✓
— **Médiane de l'attente** : 4 lancers ✓
— **Contrôle** : $\left(\frac{5}{6}\right)^{4} \approx 0{,}482 < 0{,}5$ ✓
— **Comparaison moyenne–médiane** : $6 > 4$ ✓
— **Cohérence** : signature d'une distribution étalée vers les grandes valeurs ✓
— **Programme pour estimer la moyenne** : appeler *attente()* mille fois et faire la moyenne ✓
— **Résultat attendu** : une valeur proche de 6, avec une fluctuation de l'ordre de 0,17 ✓

Réponse. La moyenne attendue est $\frac{1}{1/6} = 6$ lancers (mais la médiane n'est que de 4).

Correction de l'exercice 16 – Synthèse : simuler un contrôle qualité

a) **Simulation d'un lot** ✓

```
from random import random

def lot(n, p):
    d = 0
    for i in range(n):
        if random() < p:
            d = d + 1
    return d
```

— **Appel pour notre cas** : *lot(50, 0.04)* ✓

b) **Moyenne théorique** : $50 \times 0{,}04$ ✓

— **Résultat** :

2 défectueuses par lot

— **Interprétation** : en moyenne 2, mais le nombre **varie** d'un lot à l'autre ✓
— **Écart-type théorique** : $\sqrt{50 \times 0{,}04 \times 0{,}96} \approx 1{,}386$ ✓
— **Fourchette usuelle** : de 0 à 5 défectueuses ✓

c) **Programme d'estimation** ✓

```
def probaRejet(repetitions, n, p, seuil):
    c = 0
    for k in range(repetitions):
        if lot(n, p) >= seuil:
            c = c + 1
    return c / repetitions
```

— **Appel** : *probaRejet(100000, 50, 0.04, 4)* ✓
— **Calcul théorique** : on passe par le contraire, au plus 3 défectueuses ✓
— **Probabilité de** 0 : $0{,}96^{50} \approx 0{,}1299$ ✓
— **Probabilité de** 1 : $50 \times 0{,}04 \times 0{,}96^{49} \approx 0{,}2706$ ✓
— **Probabilité de** 2 : $1225 \times 0{,}04^{2} \times 0{,}96^{48} \approx 0{,}2762$ ✓
— **Probabilité de** 3 : $19\,600 \times 0{,}04^{3} \times 0{,}96^{47} \approx 0{,}1842$ ✓
— **Somme** : $\approx 0{,}8609$ ✓
— **Probabilité de rejet** :

$$\approx 1 - 0{,}8609 = 0{,}1391$$

— **Fréquence attendue en simulation** : environ 0,139 ✓

d) **Sur** 1000 **lots à** 4 % ✓

— **Nombre de rejets attendu** :

$$0{,}1391 \times 1000 \approx 139 \text{ lots}$$

— **Fluctuation** : $\sqrt{1000 \times 0{,}1391 \times 0{,}8609} \approx 10{,}9$ ✓
— **Fourchette plausible** : environ $[117\,;161]$ ✓
— **Interprétation industrielle** : près de 14 % des lots rejetés, c'est beaucoup ✓
— **À** 8 % **de défauts** : la moyenne par lot devient 4 ✓
— **Probabilité de** 0 : $0{,}92^{50} \approx 0{,}0155$ ✓
— **Probabilité de** 1 : $50 \times 0{,}08 \times 0{,}92^{49} \approx 0{,}0672$ ✓
— **Probabilité de** 2 : $1225 \times 0{,}08^{2} \times 0{,}92^{48} \approx 0{,}1433$ ✓
— **Probabilité de** 3 : $19\,600 \times 0{,}08^{3} \times 0{,}92^{47} \approx 0{,}1993$ ✓
— **Somme** : ≈ 0,4253 ✓
— **Probabilité de rejet** : ≈ 0,5747 ✓
— **Nombre de rejets sur** 1000 **lots** :

$$\approx 575 \text{ lots}$$

— **Comparaison** : 139 contre 575, soit plus de **quatre fois** plus ✓
— **Conclusion frappante** : doubler le taux de défaut multiplie les rejets par 4,1 ✓
— **Leçon industrielle** : le contrôle par seuil est **très sensible** à la dérive du procédé ✓

Réponse. b) 2 défectueuses par lot · c) ≈ 0,139 · d) environ 139 rejets sur 1000 lots à 4 %, contre ≈ 575 à 8 %.