

Baccalauréat général – spécialité mathématiques

Corrections expliquées

Algorithmique et programmation

À consulter après une première recherche personnelle

Utilisation. Cherche d’abord dans le fascicule du cours. Lis son indice si tu es bloquée, puis compare la logique du programme, les valeurs intermédiaires et le résultat. Les exercices sont des créations pédagogiques et ne sont pas des sujets officiels.

Rappel central. Un programme correct traduit une méthode précise : entrées, initialisation, traitement, condition d’arrêt et sortie. Tester sur un cas très simple permet de repérer une erreur, mais ne suffit pas toujours à prouver la correction générale.

Table des matières

1	Diagnostic : réponses et rappels ciblés	3
2	Corrections des exercices 1 à 4	4
3	Corrections des exercices 5 à 8	5
4	Rappel différé : corrections	7

1 Diagnostic : réponses et rappels ciblés

Correction D1

On applique les affectations dans l'ordre :

$$x = 4 \longrightarrow x = 2 \times 4 = 8 \longrightarrow x = 8 - 3 = 5 \longrightarrow x = 5^2 = 25.$$

La valeur finale est donc 25. À chaque flèche, l'ancienne valeur est remplacée.

Correction D2

`x >= 5` teste si x est supérieur ou égal à 5. La réponse est vraie pour $x = 5$, $x = 6$, etc., et fausse pour $x < 5$.

Correction D3

`range(4)` produit les valeurs 0, 1, 2, 3.

La valeur 4 n'est pas incluse.

Correction D4

La fonction renvoie la valeur 7 à l'endroit où elle est appelée. Elle ne renvoie pas le texte « return 7 ».

Correction D5

Les indices Python commencent à 0. Le premier élément 12 a donc l'indice 0.

Correction D6

Si la condition d'une boucle reste toujours vraie, et si aucune instruction ne la rend fausse, la boucle peut continuer indéfiniment. Il faut vérifier qu'une variable évolue vers la condition d'arrêt.

Point de méthode

Une trace est un tableau de valeurs successives. Elle permet de distinguer une erreur de calcul dans le programme d'une erreur de conception, comme un mauvais indice ou une condition jamais modifiée.

2 Corrections des exercices 1 à 4

Correction 1

On part de $x = 4$.

instruction	valeur de x
initialisation	4
$x \leftarrow 2x$	8
$x \leftarrow x - 3$	5
$x \leftarrow x^2$	25

La valeur finale est $\boxed{25}$.

Correction 2

Une solution complète est :

```
def signe_nombre(x):
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0
```

Le premier test traite les positifs. Le second ne se fait que si le premier est faux ; le dernier cas est alors exactement $x = 0$.

Correction 3

On initialise la somme à zéro, puis on ajoute les trois éléments :

$$0 + 3 = 3, \quad 3 + (-1) = 2, \quad 2 + 4 = \boxed{6}.$$

Par exemple :

```
def somme_liste(valeurs):
    total = 0
    for valeur in valeurs:
        total = total + valeur
    return total
```

L'initialisation à zéro est adaptée car zéro est l'élément neutre de l'addition.

Correction 4

Une solution par boucle est :

```
def doubles_positifs(valeurs):
    resultat = []
    for valeur in valeurs:
        if valeur > 0:
            resultat.append(2 * valeur)
    return resultat
```

Pour $[-2, 3, 4, 0]$, seules 3 et 4 sont strictement positives, donc le résultat est $\boxed{[6, 8]}$. Le zéro est exclu par « strictement ».

3 Corrections des exercices 5 à 8

Correction 5

On teste la longueur avant de diviser :

```
def moyenne(valeurs):  
    if len(valeurs) == 0:  
        return None  
    total = 0  
    for valeur in valeurs:  
        total = total + valeur  
    return total / len(valeurs)
```

Pour la liste vide, la fonction renvoie `None`, car une division par zéro n'est pas définie. Pour `[4, 6]`, elle calcule

$$\frac{4 + 6}{2} = 5.$$

Correction 6

La boucle parcourt 1, 2, 3, 4. Le reste de la division par 2 vaut zéro pour les nombres pairs :

$$2 \% 2 = 0, \quad 4 \% 2 = 0.$$

Les valeurs ajoutées sont donc 2 et 4, et le programme renvoie `[2, 4]`. Le symbole `%` calcule le reste de la division euclidienne.

Correction 7

Une fonction possible est :

```
def premier_carre(limite):  
    n = 0  
    while n <= limite and n*n < 50:  
        n = n + 1  
    if n <= limite:  
        return n  
    return None
```

Les carrés sont $0^2, \dots, 7^2$, et $7^2 = 49 < 50$. Le rang suivant vaut 8, et $8^2 = 64 \geq 50$. Le résultat est donc `[8]`. Le test de `n <= limite` prévoit le cas où aucun rang ne convient.

Correction 8

On peut écrire :

```
from random import random  
  
def frequence_piles():  
    piles = 0  
    for i in range(1000):  
        if random() < 0.5:  
            piles = piles + 1  
    return piles / 1000
```

La valeur renvoyée est la fréquence observée de piles dans une réalisation de 1000 lancers. Elle devrait souvent être proche de 0,5, mais elle peut être 0,476, 0,503 ou une autre valeur : aucune valeur exacte n'est imposée par la simulation.

Point de méthode

Une correction de programme doit préciser ce qu'il renvoie, pas seulement fournir du code. Pour une liste, vérifier l'ordre, les valeurs retenues et le cas vide. Pour une boucle de recherche, vérifier le rang trouvé et le rang précédent.

4 Rappel différé : corrections

Correction R1

`range(3)` produit 0, 1, 2, donc

$$\text{list}(\text{range}(3)) = [0, 1, 2].$$

Correction R2

`return` transmet une valeur à l'appelant et permet de la réutiliser dans un calcul. `print` affiche une valeur à l'écran, mais ne la transmet pas comme résultat de la fonction. Une fonction qui fait seulement `print` renvoie généralement `None`.

Correction R3

Zéro est l'élément neutre de l'addition :

$$0 + a = a.$$

Une somme peut donc commencer à zéro sans modifier le résultat. Chaque élément sera ensuite ajouté exactement une fois.

Correction R4

On peut tester `moyenne([4,6])`, dont le résultat attendu est 5. On peut aussi tester une liste constante comme `[7,7,7]`, dont la moyenne doit être 7, ou la liste vide si le programme prévoit ce cas.