Bac Général Maths + NSI (Mathématiques et Numérique et sciences informatiques)

CORRIGÉ NON OFFICIEL

Proposition de corrigé rédigée par FormaV avec l'aide d'outils d'intelligence artificielle. Elle peut contenir des erreurs et ne remplace ni le corrigé ni le barème officiels de l'examen.

Les sujets officiels des sessions passées et nos sujets d'entraînement sont classés sur https://bac-maths-nsi.fr/annales-sujets-et-corriges

Document mis en ligne par FormaV®. Reproduction ou diffusion sans autorisation interdite.

BACCALAURÉAT GÉNÉRAL • CORRIGÉ DÉTAILLÉ

Numérique et sciences informatiques (NSI)

Session 2023 • Polynésie • Jour 1

Code sujet **Non indiqué sur le sujet** 3 h 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Bases de données et SQL
13 réponses • sujet p. 1–3

Exercice 2 — Processus et programmation orientée objet
5 réponses • sujet p. 4–5

Exercice 3 — Algorithmique, programmation orientée objet et méthode « diviser pour régner »
10 réponses • sujet p. 6–7

Les numéros, parties et pages renvoient au sujet original. Chaque réponse rappelle la consigne, explique la démarche et donne le résultat.

FormaV - document d'entraînement non officiel

Exercice 1

4 points

Bases de données et SQL

1.

Question (a)

Sujet p. 1

> Après le chargement de la table Equipe, expliquer pourquoi la requête INSERT INTO Equipe VALUES (11, "Toulouse", "2 rue du Nord, 40100 Dax", "05 04 03 02 01"); produit une erreur.

Dans la table fournie, l'identifiant 11 est déjà attribué à l'équipe Charnay. Or `id_equipe` est la clé primaire : ses valeurs doivent être uniques. L'insertion d'une nouvelle ligne ayant également l'identifiant 11 viole cette contrainte.

RÉPONSE À RETENIR

La requête échoue à cause du doublon de clé primaire `id_equipe = 11`.

Question (b)

Sujet p. 1

> Expliquer le choix du domaine de l'attribut telephone.

Un numéro de téléphone n'est pas une quantité sur laquelle on effectue des calculs. Le stocker comme texte permet de conserver le zéro initial et les espaces de présentation. Le domaine `VARCHAR(20)` est suffisamment long pour les numéros indiqués.

RÉPONSE À RETENIR

Le domaine `VARCHAR(20)` convient, car il conserve les chiffres, le zéro initial et les espaces sans traiter le numéro comme une valeur numérique.

Question (c)

Sujet p. 1

> Donner le résultat de la requête SELECT nom, adresse, telephone FROM Equipe WHERE id_equipe = 5;

La clause `WHERE` sélectionne la ligne dont la clé `id_equipe` vaut 5, puis `SELECT` renvoie les trois colonnes demandées.

RÉPONSE À RETENIR

nom	adresse	telephone
Lyon	451 cours d'Emile Zola, 69100 Villeurbanne	04 05 06 07 08

Question (d)

Sujet p. 1

> Donner et expliquer le résultat de la requête SELECT COUNT(*) FROM Equipe;

`COUNT(*)` compte toutes les lignes de la table, sans filtrer sur une colonne particulière. La table `Equipe` présentée contient 12 lignes.

RÉPONSE À RETENIR

COUNT(*)
12

Question (e)

Sujet p. 1

> Écrire la requête SQL permettant d'afficher les noms des équipes par ordre alphabétique.

On sélectionne la colonne `nom`, puis on utilise `ORDER BY` pour trier les noms par ordre croissant, donc alphabétique dans l'ordre usuel.

```
SELECT nom
FROM Equipe
ORDER BY nom ASC;
```

RÉPONSE À RETENIR

Pour la table fournie avant la correction de la question (f), les noms apparaissent ainsi :

nom
Angers
Basket Landes
Bourges
Charleville-Mézières
Charnay
Landerneau
Lattes Montpellier
Lyon
Roche Vendée
Saint-Amand
Tarbe
Villeneuve d'Ascq

Question (f)

Sujet p. 1

> Écrire la requête SQL permettant de corriger le nom de l'équipe dont l'id_equipe est égal à 4. Le nom correct est Tarbes.

La commande `UPDATE` modifie la ligne ciblée par `WHERE id_equipe = 4`. La clause `SET` remplace seulement la valeur du nom.

```sql
UPDATE Equipe
SET nom = 'Tarbes'
WHERE id_equipe = 4;
```

RÉPONSE À RETENIR

La ligne d'identifiant 4 porte désormais le nom **Tarbes**.

2.

Question (a)

Sujet p. 2

> Expliquer pourquoi l'attribut id_equipe de la table Joueuse a été déclaré clé étrangère.

L'attribut `Joueuse.id_equipe` indique à quelle équipe appartient chaque joueuse. Il est une clé étrangère car ses valeurs renvoient à des clés primaires de `Equipe`, précisément à `Equipe.id_equipe`. Cela permet de relier les joueuses à leur équipe et de garantir que l'équipe indiquée existe.

RÉPONSE À RETENIR

`Joueuse.id_equipe` est une clé étrangère qui référence `Equipe.id_equipe`.

Question (b)

Sujet p. 2

> On souhaite supprimer toutes les informations relatives à une équipe. Expliquer pourquoi on ne peut pas directement supprimer cette équipe dans la table Equipe.

Des lignes de la table `Joueuse` peuvent référencer l'équipe que l'on veut supprimer. La contrainte de clé étrangère interdit alors la suppression de cette ligne dans `Equipe`, car elle laisserait des références sans équipe correspondante. Il faut d'abord supprimer ou réaffecter les joueuses concernées, ou avoir défini une règle de suppression en cascade.

RÉPONSE À RETENIR

On ne peut pas supprimer directement l'équipe tant que des joueuses la référencent, sauf si une règle de suppression en cascade a été prévue.

Question (c)

Sujet p. 1, 2

> Écrire la requête SQL qui permet d'afficher les noms et les prénoms des joueuses de l'équipe d'Angers par ordre alphabétique des noms. On supposera que l'utilisateur qui écrit cette requête ne connaît pas l'identifiant de l'équipe d'Angers.

L'identifiant d'Angers n'étant pas connu, on relie `Joueuse` à `Equipe` par leurs identifiants d'équipe et on filtre sur le nom Angers. Le tri porte sur le nom de famille, stocké dans `Joueuse.nom`.

```sql
SELECT Joueuse.nom, Joueuse.prenom
FROM Joueuse
JOIN Equipe ON Joueuse.id_equipe = Equipe.id_equipe
WHERE Equipe.nom = 'Angers'
ORDER BY Joueuse.nom ASC;
```

RÉPONSE À RETENIR

Dans l'extrait de la table `Joueuse`, l'équipe d'Angers a l'identifiant 9. Les deux joueuses de cet extrait sont affichées ci-dessous par ordre alphabétique de leur nom :

nom	prenom
Sigmundova	Jodie Cornelie
Slonjsak	Iva

3.

Question (a)

Sujet p. 1, 2

> À partir de l'analyse de l'exemple du match numéro 10, qui a opposé Villeneuve d'Ascq à Bourges le 23/10/2021, proposer un schéma relationnel pour la table Match. Si des clés étrangères sont définies, préciser quelles tables et quels attributs elles référencent. Le score est de 73 points pour Villeneuve d'Ascq, à domicile, et de 78 points pour Bourges, en déplacement.

Il faut enregistrer l'identifiant et la date du match, les deux équipes dans leur rôle respectif (domicile ou déplacement) et les deux scores. Les deux identifiants d'équipe sont des clés étrangères vers `Equipe.id_equipe`. D'après la table Equipe, Villeneuve d'Ascq a l'identifiant 3 et Bourges l'identifiant 6.

RÉPONSE À RETENIR

Un schéma relationnel possible est :

Match(`id_match`: INT, `date_match`: DATE, `id_equipe_domicile`: INT, `id_equipe_exterieure`: INT, `score_domicile`: INT, `score_exterieur`: INT)

`id_match` est la clé primaire. `id_equipe_domicile` et `id_equipe_exterieure` sont des clés étrangères qui référencent chacune `Equipe.id_equipe`.

Question (b)

Sujet p. 1, 2

> Écrire la requête SQL qui permet l'insertion dans la table Match de l'enregistrement correspondant à l'exemple donné ci-dessus.

On insère une ligne en suivant l'ordre des colonnes du schéma proposé. La date est écrite au format ISO pour la valeur de type `DATE`. Villeneuve d'Ascq est l'équipe à domicile (identifiant 3), et Bourges l'équipe en déplacement (identifiant 6) ; leurs scores respectifs sont 73 et 78.

```
INSERT INTO Match
    (id_match, date_match, id_equipe_domicile, id_equipe_exterieure,
     score_domicile, score_exterieur)
VALUES
    (10, '2021-10-23', 3, 6, 73, 78);
```

RÉPONSE À RETENIR

La ligne insérée décrit le match 10 du 23/10/2021 : Villeneuve d'Ascq à domicile, 73 points, contre Bourges en déplacement, 78 points.

4.

Question (a)

Sujet p. 2, 3

> Proposer un schéma relationnel pour stocker dans la base de données les informations relatives aux statistiques des joueuses telles que présentées dans l'extrait du match numéro 53.

Chaque ligne de statistiques associe une joueuse à un match et mémorise ses points, rebonds et passes décisives. Une clé primaire composée du match et de la joueuse identifie une ligne et empêche d'enregistrer deux fois les statistiques de la même joueuse pour le même match. Les deux identifiants sont des clés étrangères vers les tables qui décrivent respectivement les matchs et les joueuses. Le nom de l'équipe peut être obtenu en reliant la joueuse à son équipe.

> **RÉPONSE À RETENIR**
>
> Un schéma possible est :
>
> **Statistique**(`id_match`: INT, `id_joueuse`: INT, `points`: INT, `rebonds`: INT, `passes_decisives`: INT)
>
> La clé primaire est le couple `(id_match, id_joueuse)`. `id_match` référence `Match.id_match` et `id_joueuse` référence `Joueuse.id_joueuse`.

Question (b)

Sujet p. 2, 3

> Écrire la requête SQL qui a été utilisée pour afficher la partie « Extrait des statistiques » de l'exemple ci-dessus.

On part de la table des statistiques du match 53, puis on joint les joueuses pour obtenir leurs noms et prénoms, et les équipes pour afficher le nom de l'équipe. Le filtre conserve seulement le match demandé. Sans clause `ORDER BY`, SQL ne garantit pas l'ordre d'affichage des lignes.

```
SELECT Equipe.nom AS equipe,
       Joueuse.nom AS nom,
       Joueuse.prenom AS prenom,
       Statistique.points AS points,
       Statistique.rebonds AS rebonds,
       Statistique.passes_decisives AS passes_decisives
FROM Statistique
JOIN Joueuse ON Statistique.id_joueuse = Joueuse.id_joueuse
JOIN Equipe ON Joueuse.id_equipe = Equipe.id_equipe
WHERE Statistique.id_match = 53;
```

RÉPONSE À RETENIR

Les six lignes de l'extrait sont :

equipe	nom	prenom	points	rebonds	passes decisives
Charleville-Mézières	Pouye	Tima	18	6	2
Charleville-Mézières	Akhator	Evelyn	15	17	0
Charleville-Mézières	Bouferra	Amel	10	3	9
Landerneau	Mane	Marie	18	2	3
Landerneau	Amukamara	Promise	12	2	5
Landerneau	Geiselsoder	Luisa	4	10	2

Ces lignes reproduisent l'extrait fourni ; sans tri explicite, leur ordre n'est pas garanti par SQL.

Exercice 2

4 points

Processus et programmation orientée objet

1.

Question (a)

Sujet p. 4

> Pour les trois processus soumis à l'ordonnanceur au même instant — PID 11, 20 et 32, de durées respectives 4, 2 et 3 cycles CPU et dans cet ordre d'arrivée — recopier et compléter la suite des PID dans l'ordre de leur exécution lorsque le quantum du tourniquet est d'un cycle CPU. La suite commence par : 11, 20, 32, 11, …

Avec un quantum d'un cycle, chaque passage du processeur consomme exactement un cycle au processus placé en tête de la file. S'il lui reste du travail, il est remis en queue ; s'il a terminé, il est retiré.

PID exécuté	Reste après le cycle	File après le passage
11	3	20, 32, 11
20	1	32, 11, 20
32	2	11, 20, 32
11	2	20, 32, 11
20	0	32, 11
32	1	11, 32
11	1	32, 11
32	0	11
11	0	file vide

La file initiale est 11 (4 cycles), 20 (2 cycles), 32 (3 cycles). La trace suit le principe du tourniquet jusqu'à ce que les trois processus aient terminé.

> **RÉPONSE À RETENIR**
>
> La suite complète est : **11, 20, 32, 11, 20, 32, 11, 32, 11**.

Question (b)

Sujet p. 4

> Donner la composition de la suite des PID lorsque le quantum du tourniquet est de deux cycles CPU.

Avec un quantum de deux cycles, le processus en tête exécute au plus deux cycles avant d'être retiré s'il est terminé, ou remis en queue sinon. On note les durées restantes entre parenthèses.

FormaV - document d'entraînement non officiel

Passage	PID exécuté pendant le passage	Reste après le passage	File ensuite
1	11, 11	11 : 2	20 (2), 32 (3), 11 (2)
2	20, 20	20 : 0	32 (3), 11 (2)
3	32, 32	32 : 1	11 (2), 32 (1)
4	11, 11	11 : 0	32 (1)
5	32	32 : 0	file vide

La liste demandée indique le PID à chaque cycle CPU : un PID apparaît donc plusieurs fois lorsqu'il utilise plusieurs cycles.

> **RÉPONSE À RETENIR**
>
> La suite est : **11, 11, 20, 20, 32, 32, 11, 11, 32**.

2.

Question (a)

Sujet p. 4

> Recopier et compléter l'instruction Python permettant de créer la liste d'attente initiale des processus du tableau précédent, avec le processus PID 11 à l'indice 0 de la liste d'attente : liste_attente = [Processus(..., ...), ..., ...].

Le constructeur reçoit d'abord le PID, puis la durée totale en cycles CPU. L'ordre des instances dans la liste respecte l'ordre d'arrivée ; le processus 11 est donc bien à l'indice 0.

> **RÉPONSE À RETENIR**
>
> ```
> liste_attente = [Processus(11, 4), Processus(20, 2), Processus(32, 3)]
> ```

Question (b)

Sujet p. 4, 5

> Recopier, sans les commentaires, et compléter les trois méthodes de la classe Processus : execute_un_cycle, qui met à jour le reste à faire après l'exécution d'un cycle ; change_etat, qui change l'état du processus avec la valeur passée en paramètre ; et est_termine, qui renvoie True si le processus est terminé, False sinon, en se basant sur le reste à faire.

À chaque cycle exécuté, `execute_un_cycle` diminue de 1 le nombre de cycles restant. L'ordonnanceur vérifie que le processus n'est pas terminé avant de lui attribuer un cycle : le reste ne devient donc pas négatif. La méthode `est_termine` teste si ce reste est nul.

RÉPONSE À RETENIR

```
def execute_un_cycle(self):
    self.reste_a_faire = self.reste_a_faire - 1

def change_etat(self, nouvel_etat):
    self.etat = nouvel_etat

def est_termine(self):
    return self.reste_a_faire == 0
```

Question (c)

Sujet p. 5

> Recopier et compléter le code manquant de la fonction tourniquet. Elle prend en paramètre une liste d'objets Processus donnés par ordre d'arrivée et un nombre entier positif correspondant au quantum. Elle renvoie la liste des PID dans l'ordre de leur exécution par le processeur.

On retire le premier processus de la file, puis on lui attribue des cycles tant qu'il reste du quantum et que le processus n'est pas terminé. Chaque cycle ajoute son PID à la liste de sortie et diminue son reste à faire. À la fin du tour, un processus non terminé passe à l'état « Suspendu » et rejoint la fin de la file ; un processus terminé passe à l'état « Terminé » et ne revient pas dans la file.

Trace avec les processus du sujet et un quantum de 2 : 11 exécute deux cycles et rejoint la file avec 2 cycles restants ; 20 exécute deux cycles et termine ; 32 exécute deux cycles et rejoint la file avec 1 cycle restant ; 11 exécute ses deux derniers cycles et termine ; 32 exécute son dernier cycle et termine. La liste renvoyée est donc celle obtenue à la question 1(b).

Pour des durées initiales positives et un quantum entier positif, la boucle extérieure se termine : chaque cycle diminue le total des cycles restant à exécuter, et chaque processus terminé est retiré définitivement de la file. Le test de fin du processus empêche également d'exécuter un cycle après sa terminaison.

RÉPONSE À RETENIR

```
def tourniquet(liste_attente, quantum):
    ordre_execution = []
    while liste_attente != []:
        processus = liste_attente.pop(0)
        processus.change_etat("En cours d'exécution")
        compteur_tourniquet = 0
        while compteur_tourniquet < quantum and not processus.est_termine():
            ordre_execution.append(processus.pid)
            processus.execute_un_cycle()
            compteur_tourniquet = compteur_tourniquet + 1
        if not processus.est_termine():
            processus.change_etat("Suspendu")
            liste_attente.append(processus)
        else:
            processus.change_etat("Terminé")
    return ordre_execution
```


FormaV - document d'entraînement non officiel

Exercice 3

4 points

Algorithmique, programmation orientée objet et méthode « diviser pour régner »

1.

Question (a)

Sujet p. 6

> Donner le rôle de l'instruction de la ligne 1 du code précédent : from math import sqrt.

L'instruction `from math import sqrt` importe la fonction `sqrt` du module standard `math`. Elle permet ensuite de l'appeler directement avec `sqrt(...)`, sans écrire `math.sqrt(...)`.

RÉPONSE À RETENIR

La fonction `sqrt` est rendue disponible pour calculer des racines carrées.

Question (b)

Sujet p. 6

> Expliquer le résultat suivant : 0.1 + 0.2 == 0.3 renvoie False.

Les nombres décimaux comme 0,1, 0,2 et 0,3 ne sont généralement pas représentés exactement en mémoire par des nombres flottants binaires. Les approximations utilisées pour 0,1 et 0,2 s'additionnent et donnent une valeur très légèrement différente de celle stockée pour 0,3. L'égalité stricte compare ces valeurs et renvoie donc `False`.

RÉPONSE À RETENIR

En Python, `0.1 + 0.2` vaut typiquement environ `0.30000000000000004`, et non exactement `0.3` ; ainsi `0.1 + 0.2 == 0.3` renvoie `False`.

Question (c)

Sujet p. 6

> Expliquer l'erreur obtenue après les instructions point_A = (3, 4), point_A[0] puis point_A[0] = 2. L'erreur indique : TypeError: 'tuple' object does not support item assignment.

`point_A` est un tuple, ici `(3, 4)`. On peut lire un élément avec son indice, d'où la valeur `3` renvoyée par `point_A[0]`. En revanche, les tuples sont immuables : on ne peut pas remplacer un de leurs éléments après leur création.

RÉPONSE À RETENIR

L'instruction `point_A[0] = 2` provoque donc l'erreur `TypeError: 'tuple' object does not support item assignment`. Pour modifier des coordonnées, il faut créer un nouveau tuple ou utiliser une structure modifiable, telle qu'une liste.

FormaV - document d'entraînement non officiel

2.

Question (a)

Sujet p. 6

> Recopier et compléter la ligne 6 du constructeur de la classe Segment.

Pour calculer la longueur du segment, on applique la formule de la distance entre les deux points. Si leurs coordonnées sont `(x1, y1)` et `(x2, y2)`, cette longueur est la racine carrée de la somme des carrés des différences d'abscisses et d'ordonnées.

RÉPONSE À RETENIR

La ligne 6 à compléter est :

```
self.longueur = sqrt((point2[0] - point1[0])**2 + (point2[1] - point1[1])**2)
```

Question (b)

Sujet p. 6, 7

> Recopier la fonction liste_segments sans les commentaires et compléter le code manquant. Elle prend en paramètre une liste de points et renvoie une liste contenant les objets Segment qu'il est possible de construire à partir de ces points. Les segments [AB] et [BA] étant considérés comme confondus, un seul objet doit être ajouté dans la liste.

Les indices valides des points vont de `0` à `n - 1`. Pour ne créer chaque segment qu'une seule fois, on choisit un premier indice `i` jusqu'à `n - 2`, puis un second indice strictement supérieur à `i`. Ainsi, chaque paire de points est parcourue une fois et les segments [AB] et [BA] ne sont pas comptés deux fois.

Avec les trois points du sujet, les paires parcourues sont les indices (0, 1), (0, 2) et (1, 2), soit respectivement les segments [AB], [AC] et [BC].

RÉPONSE À RETENIR

Voici la fonction complétée :

```
def liste_segments(liste_points):
    n = len(liste_points)
    segments = []
    for i in range(n - 1):
        for j in range(i + 1, n):
            seg = Segment(liste_points[i], liste_points[j])
            segments.append(seg)
    return segments
```

Question (c)

Sujet p. 6, 7

> Donner en fonction de n la longueur de la liste segments. Le résultat peut être laissé sous la forme d'une somme.

Pour chaque indice `i` de 0 à `n - 2`, la boucle intérieure construit un segment avec chacun des points d'indice strictement supérieur à `i`. Elle ajoute donc `n - i - 1` segments. En sommant ces nombres, on compte toutes les paires distinctes de points.

Pour les trois points du sujet, la somme donne 2 + 1 = 3 segments : [AB], [AC] et [BC].

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

La longueur de la liste est :

$$\sum_{i=0}^{n-2}(n-i-1) = 1+2+\cdots+(n-1) = \frac{n(n-1)}{2}.$$

Pour `n = 3`, elle vaut 3.

Question (d)

Sujet p. 6, 7

Donner, en fonction de n, la complexité en temps de la fonction liste_segments.

Les deux boucles parcourent toutes les paires distinctes de points : il y en a exactement $\frac{n(n-1)}{2}$. Pour chaque paire, la fonction construit un objet `Segment`, calcule une distance en un nombre constant d'opérations et ajoute l'objet à la liste. L'ajout à une liste Python est en temps constant amorti : quelques ajouts peuvent occasionnellement nécessiter un redimensionnement, mais le coût total des ajouts reste proportionnel au nombre d'objets ajoutés.

RÉPONSE À RETENIR

La complexité en temps est $\Theta(n^2)$, donc aussi $O(n^2)$. Le nombre de segments construits est $\frac{n(n-1)}{2}$. Pour les trois points du sujet, trois segments sont construits.

Question 3.

Sujet p. 6, 7

Écrire la fonction plus_court_segment qui prend en paramètre une liste d'objets Segment et renvoie l'objet Segment dont la longueur est la plus petite. Tester si le cas de base est atteint, c'est-à-dire lorsque la liste contient un seul segment ; découper la liste en deux listes de tailles égales à une unité près ; appeler récursivement la fonction pour rechercher le minimum dans chacune des deux listes ; comparer les deux valeurs récupérées et renvoyer la plus petite.

On traite d'abord le cas de base : une liste d'un seul segment contient nécessairement son segment le plus court. Sinon, on la partage en deux listes, puis on cherche récursivement le plus court segment dans chacune. Il suffit enfin de comparer les deux longueurs obtenues et de renvoyer le segment associé à la plus petite.

La fonction est à utiliser sur une liste non vide. C'est bien le cas pour la liste produite par `liste_segments` dans cet exercice : le nuage contient au moins deux points, donc il existe au moins un segment. À chaque appel non terminal, les deux sous-listes sont strictement plus petites que la liste initiale ; les appels finissent donc par atteindre le cas de base. La comparaison renvoie le minimum de chaque moitié, puis le minimum global : le segment renvoyé est donc le plus court de toute la liste.

Trace avec les trois points du sujet : la liste des segments est [AB, AC, BC]. Elle est partagée en [AB, AC] et [BC]. La première moitié est à son tour partagée en [AB] et [AC] ; leur comparaison choisit [AB], car sa longueur est $\sqrt{2}$, contre $\sqrt{61}$ pour [AC]. La seconde moitié renvoie [BC], de longueur $\sqrt{41}$. La comparaison finale choisit donc [AB].

Les fonctions de découpage renvoyant de nouvelles listes, leur coût de copie est linéaire à chaque niveau. Pour une liste de m segments, le temps total est alors $O(m \log m)$; la profondeur de récursion est $O(\log m)$.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

Une implémentation conforme à la méthode demandée est :

```
def plus_court_segment(liste_segments):
    if len(liste_segments) == 1:
        return liste_segments[0]
    gauche = moitie_gauche(liste_segments)
    droite = moitie_droite(liste_segments)
    segment_gauche = plus_court_segment(gauche)
    segment_droite = plus_court_segment(droite)
    if segment_gauche.longueur < segment_droite.longueur:
        return segment_gauche
    return segment_droite
```

En cas d'égalité des longueurs, cette version renvoie le segment de la moitié droite ; ce segment est aussi un minimum.

4.

Question (a)

Sujet p. 7

> On considère les trois points A(3 ; 4), B(2 ; 3) et C(−3 ; −1). Donner l'instruction Python permettant de construire la variable nuage_points contenant ces trois points.

Chaque point est représenté par un tuple contenant son abscisse puis son ordonnée. On place les trois tuples dans une liste, dans l'ordre A, B, C.

RÉPONSE À RETENIR

```
nuage_points = [(3, 4), (2, 3), (-3, -1)]
```

Question (b)

Sujet p. 6, 7

> En utilisant les fonctions de l'exercice, écrire les instructions Python qui affichent les coordonnées des deux points les plus proches du nuage de points nuage_points.

On construit d'abord la liste de tous les segments, puis on utilise `plus_court_segment` pour obtenir l'objet correspondant au minimum. Ses attributs `p1` et `p2` donnent les coordonnées des deux points à afficher.

Trace numérique : la longueur de [AB] vaut $\sqrt{(2-3)^2+(3-4)^2}=\sqrt{2}$. Celle de [AC] vaut $\sqrt{(-3-3)^2+(-1-4)^2}=\sqrt{61}$. Celle de [BC] vaut $\sqrt{(-3-2)^2+(-1-3)^2}=\sqrt{41}$. La plus petite est $\sqrt{2}$, donc le segment retenu relie A et B.

RÉPONSE À RETENIR

Les instructions qui affichent les coordonnées des deux points les plus proches sont :

```
segments = liste_segments(nuage_points)
segment_plus_court = plus_court_segment(segments)
print(segment_plus_court.p1, segment_plus_court.p2)
```

Avec les données du sujet, l'affichage est :

```
(3, 4) (2, 3)
```

Sujet : https://aeif.fr/wp-content/uploads/2025/07/Polynesie_mars_2023_1.pdf
© FormaV EI • Proposition de correction, distincte d'un corrigé officiel.

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.