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

Session 2023 • Polynésie • Jour 1

Code sujet **23-NSIJ1PO1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Bases de données relationnelles et langage SQL
13 réponses • sujet p. 2–5

Exercice 2 — Gestion des processus et programmation orientée objet
5 réponses • sujet p. 6–8

Exercice 3 — Algorithmique, programmation orientée objet et méthode diviser-pour-régner
10 réponses • sujet p. 9–11

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 relationnelles et langage SQL

1

Question a.

Sujet p. 2

> 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.

La table `Equipe` contient déjà une équipe dont `id_equipe` vaut 11 : Charnay. La requête tente d'insérer une nouvelle ligne avec cette même valeur. Or `id_equipe` est une clé primaire et doit donc être unique. C'est cette violation de la contrainte d'unicité qui provoque l'erreur.

> **RÉPONSE À RETENIR**
>
> L'insertion échoue parce qu'elle créerait une seconde ligne avec la clé primaire `id_equipe = 11`.

Question b.

Sujet p. 2

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

Un numéro de téléphone n'est pas une quantité sur laquelle on effectue des calculs. Le type texte `VARCHAR(20)` permet notamment de conserver les zéros initiaux et les espaces présents dans les numéros du tableau. Un type numérique ne conviendrait pas, car il pourrait supprimer les zéros initiaux.

> **RÉPONSE À RETENIR**
>
> Le domaine `VARCHAR(20)` est adapté : un numéro de téléphone est une chaîne de caractères, et non une valeur destinée au calcul.

Question c.

Sujet p. 2, 3

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

La condition `WHERE id_equipe = 5` sélectionne la ligne dont l'identifiant est 5. La clause `SELECT` ne conserve ensuite que les attributs `nom`, `adresse` et `telephone`.

> **RÉPONSE À RETENIR**
>
nom	adresse	telephone
Lyon	451 cours Emile Zola, 69100 Villeurbanne	04 05 06 07 08

Question d.

Sujet p. 2, 3

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

`COUNT(*)` compte le nombre de lignes de la table `Equipe`. Le tableau donné contient 12 lignes, correspondant aux identifiants de 1 à 12.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

COUNT(*)
12

Question e.

Sujet p. 2, 3

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

On sélectionne l'attribut `nom` et on trie les lignes selon cet attribut avec `ORDER BY`. Le tri est croissant par défaut, ce qui donne l'ordre alphabétique demandé.

RÉPONSE À RETENIR

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

Question f.

Sujet p. 2, 3

> 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 clause `WHERE` cible uniquement l'équipe dont l'identifiant vaut 4, et `SET` remplace son nom par `Tarbes`. Sans cette condition, toutes les lignes pourraient être modifiées.

RÉPONSE À RETENIR

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

2

Question a.

Sujet p. 3

> Expliquer pourquoi l'attribut id_equipe a été déclaré clé étrangère. La clé étrangère Joueuse.id_equipe fait référence à la clé primaire Equipe.id_equipe de la table Equipe.

L'attribut `Joueuse.id_equipe` indique à quelle équipe appartient chaque joueuse. Il référence la clé primaire `Equipe.id_equipe`. Cette clé étrangère permet de vérifier que l'identifiant attribué à une joueuse correspond à une équipe présente dans la table `Equipe`.

RÉPONSE À RETENIR

`Joueuse.id_equipe` est une clé étrangère vers `Equipe.id_equipe` : elle relie chaque joueuse à une équipe existante et garantit la cohérence de cette référence.

FormaV - document d'entraînement non officiel

Question b.

Sujet p. 3, 4

> 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 faire référence à l'équipe au moyen de leur clé étrangère `id_equipe`. Supprimer cette équipe alors que ces lignes existent laisserait des références vers une équipe inexistante. La contrainte de clé étrangère interdit donc la suppression directe, sauf règle particulière telle qu'une suppression en cascade. Il faut d'abord supprimer ou réaffecter les joueuses concernées.

RÉPONSE À RETENIR

Il faut d'abord traiter les lignes de `Joueuse` qui référencent l'équipe ; sinon, la contrainte de clé étrangère empêche de supprimer cette équipe de `Equipe`.

Question c.

Sujet p. 2, 3, 4

> 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 à l'avance, on relie `Joueuse` à `Equipe` par leurs attributs correspondants, puis on sélectionne l'équipe dont le nom est `Angers`. Enfin, on trie les joueuses selon leur nom. Dans l'extrait fourni, Angers a l'identifiant 9 : les joueuses associées sont Sigmundova et Slonjsak.

RÉPONSE À RETENIR

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

Résultat sur l'extrait fourni de `Joueuse` :

nom	prenom
Sigmundova	Jodie Cornelie
Slonjsak	Iva

3

Question a.

Sujet p. 4

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

Un match comporte un identifiant, une date, deux équipes et deux scores. Il faut distinguer le rôle de chaque équipe : l'une joue à domicile, l'autre à l'extérieur. Les deux attributs d'identifiant d'équipe sont donc 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, score_domicile INT, id_equipe_exterieur INT, score_exterieur INT)`
>
> - Clé primaire : `id_match`.
> - Clé étrangère : `id_equipe_domicile` référence `Equipe.id_equipe`.
> - Clé étrangère : `id_equipe_exterieur` référence `Equipe.id_equipe`.

Question b.

Sujet p. 4

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

En utilisant l'ordre des attributs du schéma proposé, on insère le numéro du match, sa date, puis l'identifiant et le score de l'équipe à domicile et ceux de l'équipe à l'extérieur. Villeneuve d'Ascq (identifiant 3) reçoit Bourges (identifiant 6).

> **RÉPONSE À RETENIR**
>
> ```
> INSERT INTO Match
> VALUES (10, '2021-10-23', 3, 73, 6, 78);
> ```

4

Question a.

Sujet p. 4, 5

> 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 n° 53 : nombre de points marqués, rebonds et passes décisives.

Chaque ligne associe les statistiques d'une joueuse à un match. Une joueuse peut participer à plusieurs matchs, donc son identifiant seul ne suffit pas à identifier une ligne de statistiques. On peut utiliser comme clé primaire composée le couple formé par l'identifiant du match et celui de la joueuse. Ces attributs référencent respectivement les clés primaires de `Match` et de `Joueuse`.

RÉPONSE À RETENIR

Un schéma relationnel possible est :

`Statistique(id_match INT, id_joueuse INT, points INT, rebonds INT, passes_decisives INT)`

- Clé primaire composée : `(id_match, id_joueuse)`.
- `id_match` est une clé étrangère vers `Match.id_match`.
- `id_joueuse` est une clé étrangère vers `Joueuse.id_joueuse`.

Question b.

Sujet p. 4, 5

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

La requête relie les statistiques aux joueuses afin d'obtenir leurs nom et prénom, puis aux équipes afin d'obtenir le nom de l'équipe. La condition sur le numéro du match conserve uniquement les statistiques du match 53. Les alias rendent distincts les deux attributs affichés sous le nom `nom`. Sans `ORDER BY`, SQL ne garantit pas l'ordre d'affichage des lignes.

RÉPONSE À RETENIR

```
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;
```

Les valeurs correspondant à l'extrait du match n° 53 sont :

Equipe	Nom	Prénom	Points	Rebonds	Passes décisives
Charleville-Mézières	Pouye	Tima	18	6	2
Charleville-Mézières	Akhator	Evelyn	15	17	0
Charleville-Mézières	Bouderra	Amel	10	3	9
Landerneau	Mane	Marie	18	2	3
Landerneau	Amukamara	Promise	12	2	5
Landerneau	Geiselsoder	Luisa	4	10	2

Exercice 2

3 points

Gestion des processus et programmation orientée objet

1

Question a.

Sujet p. 6

> Si le quantum du tourniquet est d'un cycle CPU, recopier et compléter la suite des PID des processus dans l'ordre de leur exécution : 11, 20, 32, 11, … Les processus ont pour PID, durée et ordre d'arrivée : 11, 4 cycles CPU, 1 ; 20, 2 cycles CPU, 2 ; 32, 3 cycles CPU, 3.

Avec un quantum d'un cycle, chaque passage du processus sur le processeur correspond à une seule occurrence de son PID dans la suite. La file initiale est `[11, 20, 32]`. Après les quatre premiers cycles indiqués dans le sujet, les processus ont respectivement 2, 1 et 2 cycles à effectuer ; la file est alors `[20, 32, 11]`. On poursuit selon le même principe jusqu'à ce que chaque processus ait terminé sa durée totale.

Trace avec un quantum d'un cycle CPU

Cycle	PID exécuté	Reste après exécution
1	11	11 : 3 ; 20 : 2 ; 32 : 3
2	20	11 : 3 ; 20 : 1 ; 32 : 3
3	32	11 : 3 ; 20 : 1 ; 32 : 2
4	11	11 : 2 ; 20 : 1 ; 32 : 2
5	20	11 : 2 ; 32 : 2
6	32	11 : 2 ; 32 : 1
7	11	11 : 1 ; 32 : 1
8	32	11 : 1
9	11	aucun processus restant

Un processus dont le reste est nul est terminé et sort de la file.

> **RÉPONSE À RETENIR**
>
> La suite complète des PID dans l'ordre d'exécution est : **11, 20, 32, 11, 20, 32, 11, 32, 11**.

Question b.

Sujet p. 6

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

Avec un quantum de deux cycles CPU, un processus exécute jusqu'à deux cycles lors de chaque passage. S'il lui reste du travail, il est replacé à la fin de la file ; s'il a terminé, il en sort. Chaque cycle exécuté ajoute une occurrence du PID à la suite.

- Le PID 11 exécute deux cycles et il lui en reste 2.
- Le PID 20 exécute deux cycles et termine.

FormaV - document d'entraînement non officiel

- Le PID 32 exécute deux cycles et il lui en reste 1.
- Le PID 11 exécute ses deux derniers cycles et termine.
- Le PID 32 exécute son dernier cycle et termine.

Trace avec un quantum de deux cycles CPU

Passage	PID	Cycles exécutés	Reste après le passage	État
1	11	2	2	Suspendu, replacé en fin de file
2	20	2	0	Terminé
3	32	2	1	Suspendu, replacé en fin de file
4	11	2	0	Terminé
5	32	1	0	Terminé

Chaque cycle exécuté correspond à une occurrence du PID dans la suite.

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

2

Question a.

Sujet p. 6, 7

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

Les processus sont créés dans l'ordre d'arrivée. Le PID 11 doit être à l'indice 0 ; les durées correspondantes sont 4 cycles pour le PID 11, 2 pour le PID 20 et 3 pour le PID 32.

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

Question b.

Sujet p. 7

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

`execute_un_cycle` diminue d'une unité le nombre de cycles restant. `change_etat` affecte à l'attribut `etat` la valeur reçue. Puisque l'exécution décrémente le reste d'un cycle à la fois sans exécuter un processus déjà terminé, celui-ci est terminé lorsque `reste_a_faire` vaut zéro.

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. 8

> La fonction tourniquet prend en paramètres 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. Recopier et compléter sur la copie le code manquant.

La boucle intérieure s'exécute tant que le quantum n'est pas épuisé et que le processus n'est pas terminé. À chaque cycle, elle ajoute le PID à la liste des exécutions, décrémente le reste à faire et augmente le compteur. À la fin de son passage, un processus inachevé est suspendu et ajouté à la fin de la file ; un processus terminé reçoit l'état « Terminé » et sort définitivement de la file.

Avec les processus du sujet et un quantum de 2, les passages sont : PID 11 pendant 2 cycles (reste 2), PID 20 pendant 2 (terminé), PID 32 pendant 2 (reste 1), PID 11 pendant 2 (terminé), puis PID 32 pendant 1 (terminé). La liste retournée est donc celle donnée ci-dessous.

Trace de tourniquet avec un quantum de 2

Passage	PID exécuté	Cycles pendant le passage	Reste après le passage	État après le passage
1	11	2	2	Suspendu, replacé en fin de file
2	20	2	0	Terminé
3	32	2	1	Suspendu, replacé en fin de file
4	11	2	0	Terminé
5	32	1	0	Terminé

Chaque cycle de la colonne « Cycles pendant le passage » ajoute une occurrence du PID à la liste d'exécution.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
def tourniquet(liste_attente, quantum):
    ordre_execution = []
    while liste_attente != []:
        # On extrait le premier processus
        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
```

Pour `[Processus(11, 4), Processus(20, 2), Processus(32, 3)]` avec `quantum = 2`, la valeur renvoyée est `[11, 11, 20, 20, 32, 32, 11, 11, 32]`.

Exercice 3

5 points

Algorithmique, programmation orientée objet et méthode diviser-pour-régner

1

Question a.

Sujet p. 9

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

`from math import sqrt` importe la fonction `sqrt` du module standard `math` dans l'espace de noms du programme. On peut ensuite appeler directement `sqrt(...)`, sans écrire `math.sqrt(...)`.

RÉPONSE À RETENIR

Cette instruction rend disponible la fonction de calcul de la racine carrée, notamment utilisée pour calculer la longueur d'un segment.

Question b.

Sujet p. 9

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

Les nombres décimaux tels que 0,1 et 0,2 ne sont généralement pas représentés exactement en mémoire en écriture binaire : les nombres flottants stockent des approximations. L'addition des approximations de 0,1 et 0,2 donne une valeur légèrement différente de celle stockée pour 0,3. L'opérateur == teste l'égalité exacte des valeurs stockées, sans tolérance.

RÉPONSE À RETENIR

`0.1 + 0.2 == 0.3` vaut `False` ; en Python, la somme est représentée par une valeur proche de `0.30000000000000004`, qui n'est pas égale à la représentation de `0.3`.

Question c.

Sujet p. 9

> Expliquer l'erreur produite après avoir défini point_A = (3, 4), affiché point_A[0], puis tenté d'exécuter point_A[0] = 2 : TypeError: 'tuple' object does not support item assignment.

`point_A` est un tuple. On peut accéder à son premier élément par `point_A[0]`, mais un tuple est immuable : ses éléments ne peuvent pas être modifiés par affectation. La tentative `point_A[0] = 2` déclenche donc une erreur de type.

RÉPONSE À RETENIR

`TypeError: 'tuple' object does not support item assignment` signifie qu'on ne peut pas affecter une nouvelle valeur à un élément de tuple. Pour modifier les coordonnées, il faudrait créer un nouveau tuple.

FormaV - document d'entraînement non officiel

2

Question a.

Sujet p. 10

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

La longueur du segment entre `point1` et `point2` se calcule avec la formule de distance donnée dans l'énoncé. Dans chaque tuple, l'indice 0 désigne l'abscisse et l'indice 1 l'ordonnée.

RÉPONSE À RETENIR

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

Question b.

Sujet p. 10

> Recopier la fonction liste_segments sans les commentaires et compléter le code manquant. La fonction 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.

On parcourt les indices `i` de 0 à `n - 2`, puis les indices `j` strictement supérieurs à `i`. Ainsi, chaque paire de points distincts est traitée une seule fois : on construit [AB], mais pas une seconde fois [BA].

RÉPONSE À RETENIR

```
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
```

Chaque objet ajouté représente le segment reliant une paire de points distincts.

Question c.

Sujet p. 10

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

Pour un indice `i`, l'indice `j` prend les valeurs de `i + 1` à `n - 1`, soit `n - i - 1` valeurs. On additionne ce nombre pour tous les indices `i` de 0 à `n - 2`. Cette somme compte toutes les paires de points distincts sans distinguer leur ordre.

RÉPONSE À RETENIR

La longueur de `segments` est $\sum_{i=0}^{n-2}(n-i-1)\breve{0}000 = \frac{n(n-1)}{2}$.

FormaV - document d'entraînement non officiel

Question d.

Sujet p. 10

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

La double boucle traite toutes les paires d'indices `i < j`. Le nombre d'itérations est donc $\frac{n(n-1)}{2}$ŏ000. À chaque itération, la création du segment et le calcul de sa longueur demandent un nombre constant d'opérations.

> **RÉPONSE À RETENIR**
>
> La complexité temporelle de `liste_segments` est quadratique : $\Theta(n^2)$ŏ000, donc également $O(n^2)$ŏ000 .

3

Question a.

Sujet p. 10, 11

> 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. Procéder ainsi : 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 des deux.

Si la liste contient un seul segment, celui-ci est nécessairement le plus court : c'est le cas de base. Sinon, les fonctions fournies découpent la liste en deux listes non vides dont les tailles diffèrent d'au plus une unité. On recherche récursivement le plus court segment dans chacune, puis on compare leurs longueurs. La fonction termine car chaque sous-liste est strictement plus petite que la liste de départ, jusqu'à obtenir des listes de longueur 1. La fonction est appelée ici sur une liste non vide, puisque le nuage contient au moins deux points.

> **RÉPONSE À RETENIR**
>
> ```
> def plus_court_segment(segments):
>     if len(segments) == 1:
>         return segments[0]
>
>     gauche = moitie_gauche(segments)
>     droite = moitie_droite(segments)
>
>     minimum_gauche = plus_court_segment(gauche)
>     minimum_droite = plus_court_segment(droite)
>
>     if minimum_gauche.longueur <= minimum_droite.longueur:
>         return minimum_gauche
>     else:
>         return minimum_droite
> ```
>
> En cas d'égalité, cette version renvoie le segment retenu dans la partie gauche.

FormaV - document d'entraînement non officiel

4

Question a.

Sujet p. 11

> 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 les trois points A, B et C.

Chaque point est représenté par un tuple dont le premier élément est l'abscisse et le second l'ordonnée. On place les trois tuples dans une liste, dans l'ordre A, B, C indiqué par l'énoncé.

RÉPONSE À RETENIR

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

Question b.

Sujet p. 10, 11

> 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 tous les segments entre paires de points, puis on recherche le segment de longueur minimale et on affiche ses deux extrémités. Pour vérifier le résultat avec les données de l'énoncé, la liste des segments est, dans l'ordre, AB, AC et BC. Leurs longueurs sont respectivement $\sqrt{(2-3)^2+(3-4)^2}=\sqrt{2}$, $\sqrt{(-3-3)^2+(-1-4)^2}=\sqrt{61}$ et $\sqrt{(-3-2)^2+(-1-3)^2}=\sqrt{41}$. La fonction récursive compare AB et AC dans la moitié gauche, puis compare le plus court obtenu avec BC dans la moitié droite : le minimum est AB.

RÉPONSE À RETENIR

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

L'affichage est `(3, 4) (2, 3)`. Les deux points les plus proches sont A et B, à la distance $\sqrt{2}$.

Sujet : https://www.sujetdebac.fr/annales-pdf/2023/spe-numerique-informatique-2023-polynesie-1-sujet-officiel.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.