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 2024 • Amérique du Nord • Jour 2

Code sujet **24-NSIJ2AN1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Algorithmes de tri : tri de Stooge et programmation en Python
8 réponses • sujet p. 2–4

Exercice 2 — Langage SQL et bases de données : gestion d’une officine
8 réponses • sujet p. 5–7

Exercice 3 — Programmation objet, structures de données, réseaux et architecture matérielle : blockchain
14 réponses • sujet p. 8–12

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

6 points

Algorithmes de tri : tri de Stooge et programmation en Python

Question 1.

Sujet p. 2

> Écrire la fonction echange(tab, i, j), qui prend en arguments une liste Python tab et deux indices i et j, réalise sur place l'échange des valeurs dans tab à ces indices et ne renvoie rien.

On peut échanger les deux valeurs avec l'affectation multiple de Python. Comme on modifie les cases de `tab` directement et qu'il n'y a pas d'instruction `return`, la fonction ne renvoie rien explicitement.

```
def echange(tab, i, j):
    tab[i], tab[j] = tab[j], tab[i]
```

RÉPONSE À RETENIR

Cette fonction échange sur place les valeurs de `tab[i]` et `tab[j]`.

Question 2.

Sujet p. 2, 3

> Finaliser le programme précédent en complétant les lignes 6, 7 et 8 sur votre copie.

Les bornes des intervalles sont inclusives. Les deux premiers tiers vont donc de `i` à `j-k`, et les deux derniers tiers de `i+k` à `j`. Le troisième appel trie une nouvelle fois les deux premiers tiers.

```
        triStooge(tab, i, j-k)
        triStooge(tab, i+k, j)
        triStooge(tab, i, j-k)
```

RÉPONSE À RETENIR

Les lignes 6, 7 et 8 sont respectivement `triStooge(tab, i, j-k)`, `triStooge(tab, i+k, j)` et `triStooge(tab, i, j-k)`.

Question 3.

Sujet p. 3

> Indiquer, en le justifiant, si cet algorithme est itératif ou récursif.

Un algorithme est récursif si une fonction s'appelle elle-même. Ici, `triStooge` s'appelle sur plusieurs sous-intervalles du tableau. Les appels s'arrêtent lorsque `j - i` ne dépasse pas 1, car la condition de récursion de la ligne 4 n'est alors plus vérifiée.

RÉPONSE À RETENIR

L'algorithme est **récursif**, car `triStooge` s'appelle elle-même.

Question 4.

Sujet p. 2, 3

> Soit l'appel triStooge(A, 0, 5) avec A = [5, 6, 4, 2, 3, 1]. Déterminer la valeur numérique prise par k lors de ce

FormaV - document d'entraînement non officiel

> premier appel. Une justification est attendue.

Pour le premier appel, `i = 0` et `j = 5`. En appliquant la formule de l'énoncé, on obtient `k = (j - i + 1) // 3 = (5 - 0 + 1) // 3 = 6 // 3`.

> **RÉPONSE À RETENIR**
>
> La valeur prise par `k` lors du premier appel est **2**.

Question 5.

Sujet p. 3

> À partir de l'arbre des appels récursifs effectués depuis l'appel triStooge(A, 0, 5), dénombrer le nombre d'appels récursifs effectués lors du tri sans compter l'appel initial.

L'appel initial produit trois appels sur des intervalles de quatre éléments. Chacun de ces trois appels produit trois appels sur des intervalles de trois éléments : cela fait `3 × 3 = 9` appels supplémentaires. Chacun de ces neuf appels produit à son tour trois appels terminaux, soit `9 × 3 = 27` appels. On ne compte pas l'appel initial lui-même.

> **RÉPONSE À RETENIR**
>
> Le nombre d'appels récursifs sans compter l'appel initial est `3 + 9 + 27 = 39`.

Question 6.

Sujet p. 3

> Déterminer les appels effectués dans les cases 1, 2 et 3 de l'arbre des appels récursifs.

On applique à chaque intervalle la formule de découpage, avec des bornes inclusives.

- La case 1 est le deuxième sous-appel de `triStooge(A, 0, 3)`. Pour cet intervalle, `k = (3 - 0 + 1) // 3 = 1`, donc le deuxième sous-appel porte sur les indices `i+k` à `j`, soit `1` à `3`.
- La case 2 est le premier sous-appel de `triStooge(A, 2, 4)`. Ici `k = (4 - 2 + 1) // 3 = 1`, donc ce sous-appel porte sur les indices `2` à `3`.
- La case 3 correspond au troisième sous-appel de l'appel initial. Il trie de nouveau les deux premiers tiers, comme le premier sous-appel, donc de l'indice `0` à l'indice `3`.

> **RÉPONSE À RETENIR**
>
> Case 1 : `triStooge(A, 1, 3)` ; case 2 : `triStooge(A, 2, 3)` ; case 3 : `triStooge(A, 0, 3)`.

Question 7.

Sujet p. 3, 4

> Dans cette question, A = [5, 4, 6, 2]. Recopier le tableau et le compléter en remplaçant les ?? : triStooge(A, 0, 3), avec A avant l'appel égal à [5, 6, 4, 2] ; un appel manquant avec A avant l'appel égal à [2, 6, 4, 5] ; triStooge(A, 1, 3), avec A avant l'appel égal à [2, 4, 6, 5] ; puis un appel manquant avec A avant et après l'appel égal à [2, 4, 5, 6]. L'énoncé indique A = [5, 4, 6, 2], tandis que la première ligne du tableau indique [5, 6, 4, 2].

Il y a une incohérence dans l'énoncé : il annonce `A = [5, 4, 6, 2]`, alors que la première ligne du tableau donne `[5, 6, 4, 2]`. Pour remplir le tableau, on suit les valeurs explicitement indiquées dans ses lignes.

Dans `triStooge(A, 0, 3)`, les valeurs aux indices 0 et 3 sont échangées : le tableau devient `[2, 6, 4, 5]`. Le premier sous-appel est `triStooge(A, 0, 2)`. Il trie la partie allant des indices 0 à 2 et donne `[2, 4, 6, 5]`. L'appel ensuite indiqué, `triStooge(A, 1, 3)`, donne `[2, 4, 5, 6]`. Enfin, le dernier sous-

appel est de nouveau `triStooge(A, 0, 2)` : cette partie étant déjà triée, le tableau ne change pas.

RÉPONSE À RETENIR

Appel	Valeur de A avant l'appel	Valeur de A après l'appel
`triStooge(A, 0, 3)`	`[5, 6, 4, 2]`	`[2, 4, 5, 6]`
`triStooge(A, 0, 2)`	`[2, 6, 4, 5]`	`[2, 4, 6, 5]`
`triStooge(A, 1, 3)`	`[2, 4, 6, 5]`	`[2, 4, 5, 6]`
`triStooge(A, 0, 2)`	`[2, 4, 5, 6]`	`[2, 4, 5, 6]`

Question 8.

Sujet p. 4

On montre que le coût en temps dans le pire des cas de l'algorithme de Stooge est de l'ordre de n^e, avec e environ égal à 8/3. Donner un algorithme de tri dont le coût est strictement meilleur.

Le tri par insertion a un coût en temps de l'ordre de n^2 dans le pire des cas. Pour les tailles n suffisamment grandes, n^2 croît strictement moins vite que $n^{8/3}$, puisque $2 < 8/3$. C'est donc un algorithme de tri dont le coût est strictement meilleur que celui du tri de Stooge indiqué dans l'énoncé.

RÉPONSE À RETENIR

Par exemple, le **tri par insertion**, de coût en temps $O(n^2)$ dans le pire des cas, est strictement meilleur que le tri de Stooge, de coût de l'ordre de $n^{8/3}$.

Exercice 2

6 points

Langage SQL et bases de données : gestion d'une officine

Question 1.

Sujet p. 5

> Pour la relation client présentée, écrire le résultat de l'exécution de la requête SQL : SELECT nom_client, prenom_client FROM client ORDER BY nom_client;

`ORDER BY nom_client` classe les lignes par ordre alphabétique croissant des noms. « Dufour » précède « Martin » ; la requête n'affiche que les colonnes `nom_client` et `prenom_client`.

RÉPONSE À RETENIR

nom_client	prenom_client
Dufour	Marc
Martin	Sophie

Question 2.

Sujet p. 6

> Écrire une requête SQL permettant d'afficher les noms de tous les médicaments dont le prix est strictement inférieur à 3 euros.

On sélectionne le nom des médicaments dans `medicament` et on filtre les lignes avec la condition stricte `prix < 3`. Dans la table, les deux médicaments concernés coûtent 2,30 euros.

```
SELECT nom_medic
FROM medicament
WHERE prix < 3;
```

Sans clause `ORDER BY`, l'ordre d'affichage des résultats n'est pas garanti.

RÉPONSE À RETENIR

nom_medic
Acide acétylsalicylique
Gel hydroalcoolique 100 ml

Question 3.

Sujet p. 5, 7

> Écrire une requête SQL permettant d'ajouter les informations de la carte vitale de la troisième cliente présentée : Nathalie Durand, numéro de sécurité sociale 2 69 05 49 588 157 80.

La carte indique le nom Durand, le prénom Nathalie et le numéro de sécurité sociale `2 69 05 49 588 157 80`. Comme l'attribut est de type `VARCHAR(15)`, on enregistre le numéro comme une chaîne, sans espaces. Les identifiants déjà utilisés étant 1 et 2, on choisit 3, en supposant qu'il est disponible.

FormaV - document d'entraînement non officiel

```
INSERT INTO client (id_client, nom_client, prenom_client, num_secu_sociale)
VALUES (3, 'Durand', 'Nathalie', '269054958815780');
```

RÉPONSE À RETENIR

La requête ajoute Nathalie Durand avec l'identifiant 3 et le numéro de sécurité sociale `269054958815780`.

Question 4.

Sujet p. 5, 6, 7

> Donner les attributs qui doivent être déclarés comme clés étrangères de la relation ordonnance et en préciser l'utilité.

La relation `ordonnance` contient les identifiants du client concerné et du médicament prescrit. Ces attributs doivent référencer les identifiants correspondants dans les tables où ils sont définis comme clés primaires.

- `ordonnance.id_client` référence `client.id_client` ;
- `ordonnance.id_medic` référence `medicament.id_medic`.

Ces clés étrangères assurent la cohérence référentielle : une ordonnance ne peut ainsi être associée qu'à un client et à un médicament existants.

RÉPONSE À RETENIR

Les clés étrangères sont `id_client` vers `client(id_client)` et `id_medic` vers `medicament(id_medic)`.

Question 5.

Sujet p. 6, 7

> Indiquer, pour les lignes 7 et 8 de la table ordonnance, le nombre de boîtes prescrites.

Pour le paracétamol, la prescription autorise au maximum 3 comprimés par jour pendant 2 jours, soit au plus `3 × 2 = 6` comprimés. Une boîte en contient 8 : une boîte suffit.

Pour l'acide ascorbique, il faut un comprimé par jour pendant 4 semaines, soit `7 × 4 = 28` comprimés. Les boîtes en contiennent 10 ; il faut donc arrondir `28 / 10 = 2,8` au nombre entier supérieur, soit 3 boîtes.

RÉPONSE À RETENIR

Ligne 7 : **1 boîte** de paracétamol. Ligne 8 : **3 boîtes** d'acide ascorbique.

Question 6.

Sujet p. 6, 7

> Écrire la requête SQL mettant à jour la quantité du médicament Acide ascorbique en stock dans l'officine du pharmacien suite au passage de Madame Martin.

Madame Martin reçoit 3 boîtes d'acide ascorbique. L'attribut `quantite` n'a pas d'unité précisée dans l'énoncé. Si cette quantité est comptée en boîtes, on retranche 3 à la quantité de l'enregistrement d'identifiant 4 :

```
UPDATE medicament
SET quantite = quantite - 3
WHERE id_medic = 4;
```


FormaV - document d'entraînement non officiel

Avec cette convention, le stock passe de 450 à `450 - 3 = 447` boîtes. Si `quantite` désigne plutôt un nombre de comprimés, les 3 boîtes contenant chacune 10 comprimés, il faut retrancher 30 :

```
UPDATE medicament
SET quantite = quantite - 30
WHERE id_medic = 4;
```

RÉPONSE À RETENIR

La mise à jour est de **450 à 447** si le stock est compté en boîtes, ou de **450 à 420** s'il est compté en comprimés. L'énoncé ne précisant pas l'unité de `quantite`, le choix entre ces deux requêtes dépend de cette convention.

Question 7.

Sujet p. 6, 7

> Calculer le coût total des médicaments fournis à Madame Martin en justifiant le calcul. Il n'est pas demandé d'écrire une requête.

La délivrance correspond à 1 boîte de paracétamol et 3 boîtes d'acide ascorbique. En considérant les prix de la table comme des prix par boîte, le calcul est :

`1 × 3,50 + 3 × 5,50 = 3,50 + 16,50 = 20,00` euros.

RÉPONSE À RETENIR

Le coût total des médicaments fournis à Madame Martin est de **20,00 euros**.

Question 8.

Sujet p. 6, 7

> Écrire la requête SQL permettant d'afficher le nom du médicament pour l'ordonnance ayant l'id_ordo numéro 6.

La ligne d'ordonnance d'identifiant 6 indique `id_medic = 2`. On joint donc cette ligne à `medicament` sur l'identifiant du médicament, puis on filtre sur l'identifiant de l'ordonnance :

```
SELECT medicament.nom_medic
FROM ordonnance
JOIN medicament ON ordonnance.id_medic = medicament.id_medic
WHERE ordonnance.id_ordo = 6;
```

RÉPONSE À RETENIR

nom_medic
Acide acétylsalicylique

FormaV - document d'entraînement non officiel

Exercice 3

8 points

Programmation objet, structures de données, réseaux et architecture matérielle : blockchain

Partie A

Question 1.

Sujet p. 8

> Dans le réseau local de masque 255.255.255.0, Alice a pour adresse IP 192.168.1.1 et Bob 192.168.1.2. L'adresse de diffusion 192.168.1.255 est réservée et ne peut être attribuée à une machine. Donner une adresse IP possible pour Charlie afin qu'il puisse communiquer avec Alice et Bob dans le réseau local. Justifier votre réponse en donnant toutes les conditions à respecter dans le choix de cette adresse IP.

Avec le masque `255.255.255.0`, les trois premiers octets identifient le réseau local. L'adresse de Charlie doit donc commencer par `192.168.1.`. Le dernier octet ne peut pas être `0`, qui désigne l'adresse du réseau, ni `255`, qui est l'adresse de diffusion. Il doit aussi être différent de `1` et `2`, déjà attribués à Alice et Bob.

Les adresses utilisables pour Charlie sont donc `192.168.1.3` à `192.168.1.254`.

RÉPONSE À RETENIR

Par exemple, Charlie peut avoir l'adresse IP **`192.168.1.3`**.

Question 2.

Sujet p. 8

> Toutes les dix minutes, les transactions réalisées pendant cet intervalle sont regroupées par ordre d'apparition dans une liste Python. Dans un intervalle de dix minutes, Alice envoie dix nsicoin à Charlie puis Bob envoie cinq nsicoin à Alice. Écrire la liste Python correspondante à ces transactions.

Le constructeur `Transaction` attend, dans l'ordre, l'expéditeur, le destinataire et le montant. La liste conserve l'ordre d'apparition des deux transactions : Alice envoie d'abord 10 nsicoin à Charlie, puis Bob envoie 5 nsicoin à Alice.

RÉPONSE À RETENIR

```
liste_transactions = [
    Transaction('Alice', 'Charlie', 10),
    Transaction('Bob', 'Alice', 5)
]
```

Question 3.

Sujet p. 9

> La figure 1 représente les trois premiers blocs d'une blockchain. Expliquer pourquoi la valeur de l'attribut bloc_precedent du bloc0 est None.

Le bloc0 est le premier bloc de la blockchain : aucun bloc ne le précède. L'attribut `bloc_precedent` doit donc représenter l'absence de bloc précédent. En Python, cette absence est représentée par la valeur `None`.

RÉPONSE À RETENIR

`bloc0.bloc_precedent` vaut **`None`**.

FormaV - document d'entraînement non officiel

Question 4.

Sujet p. 9

> Donner la valeur de l'attribut bloc_precedent du bloc1 afin que celui-ci soit lié au bloc0.

L'attribut `bloc_precedent` contient une référence vers l'objet `Bloc` précédent. Pour relier bloc1 à bloc0, on affecte donc l'objet `bloc0` à cet attribut.

RÉPONSE À RETENIR

```
bloc1.bloc_precedent = bloc0
```

Question 5.

Sujet p. 8, 9

> À l'aide des classes Bloc et Blockchain, écrire le code Python permettant de créer un objet ma_blockchain de type Blockchain représenté par la figure 1.

La construction commence par l'objet `Blockchain`, dont la tête est le bloc initial bloc0. On crée ensuite bloc1 en lui donnant bloc0 comme bloc précédent, puis bloc2 en lui donnant bloc1 comme bloc précédent. Comme la tête désigne le bloc le plus récent, elle doit finalement référencer bloc2.

Le code imprimé page 9 comporte une coquille à la ligne 10 : pour que l'appel de la ligne 8 soit cohérent et que la définition soit valide en Python, il faut écrire `def creer_bloc_0(self):`, et non `def self.creer_bloc_0(self):`. Le code ci-dessous suppose cette correction.

RÉPONSE À RETENIR

```
ma_blockchain = Blockchain()
bloc0 = ma_blockchain.tete

bloc1 = Bloc([
    Transaction('Alice', 'Charlie', 50),
    Transaction('Charlie', 'Bob', 30)
], bloc0)

bloc2 = Bloc([
    Transaction('Bob', 'Charlie', 20),
    Transaction('Bob', 'Charlie', 20),
    Transaction('Charlie', 'Alice', 30)
], bloc1)

ma_blockchain.tete = bloc2
```

Question 6.

Sujet p. 9

> Donner le solde en nsicoin de Bob à l'issue du bloc2.

Bob reçoit 100 nsicoin dans bloc0. Dans bloc1, il reçoit 30 nsicoin de Charlie. Dans bloc2, il envoie deux fois 20 nsicoin à Charlie, soit 40 nsicoin au total. La transaction de Charlie vers Alice ne change pas son solde.

Étape	Calcul du solde de Bob	Solde
Après bloc0	100	100 nsicoin
Après bloc1	100 + 30	130 nsicoin
Après bloc2	130 − 20 − 20	90 nsicoin

RÉPONSE À RETENIR

À l'issue de bloc2, le solde de Bob est de **90 nsicoin**.

Question 7.

Sujet p. 9, 10

> On souhaite doter la classe Blockchain d'une méthode ajouter_bloc qui prend en paramètre la liste des transactions des dix dernières minutes et l'ajoute dans un nouveau bloc. Écrire le code Python de cette méthode.

Le bloc courant est celui référencé par `self.tete`. Il faut créer un nouveau bloc contenant les transactions reçues et dont le bloc précédent est l'ancienne tête, puis remplacer la tête par ce nouveau bloc. Ainsi, le chaînage conserve tous les blocs antérieurs.

RÉPONSE À RETENIR

```
def ajouter_bloc(self, liste_transactions):
    nouveau_bloc = Bloc(liste_transactions, self.tete)
    self.tete = nouveau_bloc
```

Par exemple, avec la liste de transactions de la question 2, le nouveau bloc aurait cette liste comme contenu et l'ancienne valeur de `self.tete` comme `bloc_precedent`.

Question 8.

Sujet p. 8, 10

> Lorsqu'un utilisateur ajoute un nouveau bloc à la blockchain, il l'envoie aux autres membres, qui disposent chacun sur leur propre machine d'une copie identique de la blockchain. Donner le nom et la valeur de l'adresse IP à utiliser pour effectuer cet envoi.

Pour envoyer le bloc à toutes les machines du réseau local, on utilise l'adresse de diffusion (broadcast), qui permet de communiquer avec toutes les machines du réseau. Le sujet donne cette adresse et précise qu'elle est réservée à cet usage.

RÉPONSE À RETENIR

Le nom est **adresse de diffusion** (ou adresse broadcast) et sa valeur est **`192.168.1.255`**.

Question 9.

Sujet p. 9, 10

> On souhaite doter la classe Bloc d'une nouvelle méthode calculer_solde permettant de renvoyer le solde à l'issue de ce bloc. Recopier et compléter le code Python fourni pour cette méthode.

Le calcul remonte récursivement jusqu'au premier bloc, où le solde initial est nul, puis traite les transactions de chaque bloc en revenant vers le bloc courant. Une transaction diminue le solde de l'utilisateur si celui-ci est l'expéditeur et l'augmente s'il en est le destinataire. Dans le modèle donné, les attributs sont

`self.bloc_precedent` et `self.liste_transactions` ; ces noms doivent être employés à la place des noms incohérents figurant dans le code à compléter.

Par exemple, pour Alice sur la chaîne de la figure : bloc0 donne 100, bloc1 retire 50, et bloc2 ajoute 30. Son solde final est donc `100 − 50 + 30 = 80` nsicoin.

RÉPONSE À RETENIR

```
def calculer_solde(self, utilisateur):
    if self.bloc_precedent is None:
        solde = 0
    else:
        solde = self.bloc_precedent.calculer_solde(utilisateur)

    for transaction in self.liste_transactions:
        if transaction.expediteur == utilisateur:
            solde = solde - transaction.montant
        elif transaction.destinataire == utilisateur:
            solde = solde + transaction.montant

    return solde
```

Question 10.

Sujet p. 9, 10

> Écrire l'appel à la fonction calculer_solde permettant de calculer le solde actuel d'Alice.

La tête de `ma_blockchain` est le bloc le plus récent, bloc2. On appelle donc la méthode sur cette tête et on lui passe le nom d'Alice. En suivant les transactions de la figure, Alice reçoit d'abord 100 nsicoin dans bloc0, en envoie 50 dans bloc1, puis reçoit 30 dans bloc2 : `100 − 50 + 30 = 80`.

RÉPONSE À RETENIR

```
ma_blockchain.tete.calculer_solde('Alice')
```

L'appel renvoie **80 nsicoin**.

Partie B

Question 11.

Sujet p. 11

> L'attribut nonce est de type entier. Miner un bloc signifie trouver une valeur de nonce telle que l'attribut hash du bloc commence par les deux caractères « 00 ». Compte tenu des propriétés précédentes, la seule façon de trouver cette valeur est de procéder à une recherche exhaustive. Expliquer en quoi consiste le fait de trouver une valeur par recherche exhaustive.

Une recherche exhaustive consiste à essayer systématiquement les valeurs du nonce, sans en omettre, jusqu'à obtenir un hash répondant au critère. Pour chaque valeur testée, on calcule le hash du bloc puis on vérifie s'il commence par `00`. Si ce n'est pas le cas, on essaie la valeur suivante.

RÉPONSE À RETENIR

On teste successivement les nonces `0`, `1`, `2`, etc., et on s'arrête dès qu'un hash commence par `00`.

Question 12.

Sujet p. 11, 12

> En justifiant votre réponse, donner la valeur de l'attribut hash_bloc_precedent du bloc0.

Le bloc0 n'a pas de bloc précédent. La méthode `donner_hash_precedent` renvoie alors la chaîne de caractères `'0'`. Il faut donc distinguer cette valeur conventionnelle de la valeur Python `None`, qui est celle de l'attribut `bloc_precedent`.

RÉPONSE À RETENIR

`bloc0.hash_bloc_precedent` vaut **`'0'`**.

Question 13.

Sujet p. 12

> Sachant que le hash est écrit sur 256 bits, donner le calcul permettant d'obtenir le nombre de hash possibles.

Chaque bit du hash peut prendre deux valeurs, 0 ou 1. Les 256 bits peuvent être choisis indépendamment : le nombre de suites possibles est donc le produit de 256 facteurs égaux à 2.

RÉPONSE À RETENIR

Le nombre de hash possibles est 2^{256}.

Question 14.

Sujet p. 12

> Recopier et compléter sur votre copie le code Python de la méthode minage_bloc. La méthode énumère tous les entiers naturels en partant de 0 pour modifier le nonce jusqu'à ce que le hash commence par « 00 ».

On initialise le nonce à 0, puis on calcule le hash correspondant. Tant que ce hash ne commence pas par `00`, on incrémente le nonce et on recalcule le hash. La méthode modifie les attributs du bloc ; aucun résultat numérique précis ne peut être donné sans le code de `calculer_hash`.

Trace symbolique : on calcule d'abord le hash pour le nonce 0 ; s'il ne commence pas par `00`, on passe au nonce 1 et on recommence, puis on continue ainsi jusqu'au premier hash qui satisfait le critère.

RÉPONSE À RETENIR

```
def minage_bloc(self):
    self.nonce = 0
    self.hash = self.calculer_hash()
    while not self.hash.startswith('00'):
        self.nonce = self.nonce + 1
        self.hash = self.calculer_hash()
```

Sujet : https://www.sujetdebac.fr/annales-pdf/2024/spe-numerique-informatique-2024-amerique-nord-2-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.