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 • Métropole • Jour 1 • Remplacement

Code sujet **24-NSIJ1ME3** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Programmation Python, bases de données relationnelles et requêtes SQL
9 réponses • sujet p. 2–5

Exercice 2 — Réseaux, protocoles de routage et graphes
10 réponses • sujet p. 6–10

Exercice 3 — Bases de numération, pile et programmation orientée objet
12 réponses • sujet p. 11–16

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

Programmation Python, bases de données relationnelles et requêtes SQL

A

Question 1.

Sujet p. 2, 3

> Donner pour la table Agences un type possible pour l'attribut CP, qui indique le code postal.

Un code postal est un identifiant, pas une quantité destinée à être calculée. Le stocker comme du texte permet aussi de conserver un éventuel zéro initial.

> **RÉPONSE À RETENIR**
>
> Un type possible pour `CP` est `CHAR(5)` (ou `VARCHAR(5)`).

Question 2.

Sujet p. 2, 3

> Donner le résultat de la requête suivante pour les extraits des tables données : SELECT COUNT(DISTINCT 'Telephone') FROM 'Agences';

Il faut tenir compte de la requête telle qu'elle est imprimée : les apostrophes simples désignent normalement des chaînes de caractères, et non des noms d'attributs ou de tables. En SQL standard, `FROM 'Agences'` n'est pas une désignation valide de table : la requête imprimée n'a donc pas de résultat standard défini. Si l'intention est de compter les téléphones distincts, la requête correcte est `SELECT COUNT(DISTINCT Telephone) FROM Agences;`. Les cinq téléphones de l'extrait de la page 2 sont distincts. Dans un moteur permissif qui accepte `'Agences'` comme nom de table, `'Telephone'` reste une constante identique sur chaque ligne : le compte distinct vaut alors 1.

> **RÉPONSE À RETENIR**
>
> Pour la requête telle qu'imprimée : elle est invalide en SQL standard ; dans un moteur qui tolère le nom de table entre apostrophes, elle compte une constante et renvoie **1**. Pour la requête vraisemblablement visée, sans apostrophes autour des identifiants, le résultat est **5**.

Question 3.

Sujet p. 2, 3, 4

> Expliquer à quelle condition l'attribut Telephone pourrait servir de clé primaire pour la table Agences.

Une clé primaire doit identifier chaque ligne sans ambiguïté : sa valeur doit être unique et renseignée pour chaque agence. Dans la nouvelle table de la page 4, Licorne et Le Carosse partagent le numéro `0123456789` ; dans cet état, `Telephone` ne peut donc pas être la clé primaire.

> **RÉPONSE À RETENIR**
>
> `Telephone` pourrait servir de clé primaire seulement si chaque agence avait un numéro renseigné et différent de celui de toutes les autres agences.

FormaV - document d'entraînement non officiel

Question 4.

Sujet p. 3, 4, 5

> Décrire les attributs de la nouvelle table couple_voitures_agences, créée pour associer la table Agences à la table Voitures. La table Voitures a pour clé primaire id_voiture. Souligner une clé primaire et faire commencer une clé étrangère par un dièse (#).

L'exemple d'insertion de la page 5 montre que la table d'association comporte un attribut `id`, en plus des deux identifiants permettant de relier les tables. On décrit donc les attributs ainsi :

- **`id`** : clé primaire ;
- **`#id_agence`** : clé étrangère référençant `Agences.id_agence` ;
- **`#id_voiture`** : clé étrangère référençant `Voitures.id_voiture`.

La balise de soulignement n'étant pas nécessaire pour expliciter le schéma ici, la clé primaire est indiquée explicitement.

> **RÉPONSE À RETENIR**
>
> `couple_voitures_agences(id, id_agence, id_voiture)`, avec `id` comme clé primaire et `id_agence` et `id_voiture` comme clés étrangères.

Question 5.

Sujet p. 4, 5

> Écrire une requête qui permet d'enregistrer dans la table couple_voitures_agences le fait que le véhicule 2 est associé à l'agence Deep Place.

Dans la table `Agences` de la page 4, Deep Place a l'identifiant `5` ; le véhicule concerné a l'identifiant `2`. La requête ci-dessous suppose que l'identifiant de la table d'association est généré automatiquement ou possède une valeur par défaut. Si ce n'est pas le cas, il faut aussi fournir une valeur inutilisée pour l'attribut `id`.

```
INSERT INTO couple_voitures_agences (id_agence, id_voiture)
VALUES (5, 2);
```

> **RÉPONSE À RETENIR**
>
> La ligne associe la voiture `2` à l'agence Deep Place, d'identifiant `5`.

Question 6.

Sujet p. 4, 5

> Écrire une requête qui permet d'actualiser la table pour indiquer que le véhicule 2 se trouve maintenant à l'agence Le Carosse.

Le Carosse a pour identifiant `2` dans la nouvelle table `Agences`. On modifie donc l'identifiant d'agence associé à la voiture `2`. Cette mise à jour suppose que l'association de cette voiture existe déjà.

```
UPDATE couple_voitures_agences
SET id_agence = 2
WHERE id_voiture = 2;
```

> **RÉPONSE À RETENIR**
>
> Après la mise à jour, l'association indique que le véhicule `2` se trouve à l'agence Le Carosse.

FormaV - document d'entraînement non officiel

Question 7.

Sujet p. 3, 4, 5

> Écrire une requête qui permet d'afficher le type, la marque et le nom de l'agence pour l'ensemble des véhicules.

On relie les voitures à la table d'association par `id_voiture`, puis à l'agence par `id_agence`. Les jointures externes à gauche conservent toutes les voitures, y compris celles sans agence associée ; les colonnes de l'agence sont alors `NULL`.

```
SELECT Voitures.type, Voitures.marque, Agences.Agence
FROM Voitures
LEFT JOIN couple_voitures_agences
    ON Voitures.id_voiture = couple_voitures_agences.id_voiture
LEFT JOIN Agences
    ON couple_voitures_agences.id_agence = Agences.id_agence;
```

Les extraits du sujet ne donnent pas le contenu de `couple_voitures_agences`, donc le résultat réel complet n'est pas déterminable. À titre conditionnel, si les questions 5 et 6 ont été exécutées et qu'il n'existe aucune autre association, les dix lignes seraient celles-ci ; leur ordre n'est pas garanti sans `ORDER BY`.

type	marque	Agence
Berline	Renault	NULL
Berline	Renault	Le Carosse
Berline	Dacia	NULL
Berline	Dacia	NULL
Berline	Dacia	NULL
Berline	Dacia	NULL
Berline	Renault	NULL
Citadine	Fiat	NULL
Citadine	Fiat	NULL
Citadine	Fiat	NULL

RÉPONSE À RETENIR

La requête affiche le type, la marque et le nom de l'agence pour chaque voiture. Le résultat complet dépend des lignes présentes dans la table d'association, qui ne sont pas fournies dans les extraits.

B

Question 8.

Sujet p. 3, 4, 5

> Écrire en Python la fonction insert_voiture(liste_valeurs, id_agence). Cette fonction insère un nouveau véhicule dans une agence, prend en paramètres les valeurs des attributs de la voiture à insérer et l'entier id_agence, et renvoie un booléen de contrôle.

La liste contient les six attributs de voiture autres que l'identifiant auto-incrémenté, dans l'ordre du tableau : marque, modèle, kilométrage, nombre de places, type et carburant. Après l'insertion de la voiture, une seconde requête crée l'association avec l'agence. Comme la fonction fournie ne renvoie pas l'identifiant généré, le code récupère le plus grand `id_voiture`. Cela suppose qu'aucune autre insertion de voiture n'a lieu simultanément ; sans cette hypothèse, il faudrait un mécanisme de base de données permettant de récupérer l'identifiant de la ligne insérée. Les apostrophes des valeurs textuelles sont doublées pour former des chaînes SQL valides.

```
def insert_voiture(liste_valeurs, id_agence):
    if len(liste_valeurs) != 6:
        return False

    marque, modele, kilometrage, nombre_places, type_voiture, carburant = liste_valeurs

    if not all(isinstance(v, str) for v in
               (marque, modele, type_voiture, carburant)):
        return False
    if not isinstance(kilometrage, int) or kilometrage < 0:
        return False
    if not isinstance(nombre_places, int) or nombre_places <= 0:
        return False
    if not isinstance(id_agence, int):
        return False

    marque = marque.replace("'", "''")
    modele = modele.replace("'", "''")
    type_voiture = type_voiture.replace("'", "''")
    carburant = carburant.replace("'", "''")

    requete_voiture = (
        "INSERT INTO Voitures "
        "(marque, modele, kilometrage, nombre_places, type, carburant) "
        "VALUES ('{}', '{}', {}, {}, '{}', '{}')"
    ).format(
        marque, modele, kilometrage, nombre_places, type_voiture, carburant
    )

    if not execute_requete_insert(requete_voiture):
        return False

    requete_association = (
        "INSERT INTO couple_voitures_agences (id_agence, id_voiture) "
        "VALUES ({}, (SELECT MAX(id_voiture) FROM Voitures))"
    ).format(id_agence)

    return execute_requete_insert(requete_association)
```

Si la première insertion réussit mais que la seconde échoue, la voiture demeure enregistrée sans association. Garantir la réussite ou l'échec des deux opérations ensemble nécessiterait une transaction, qui n'est pas fournie dans l'énoncé.

> **RÉPONSE À RETENIR**
>
> La fonction renvoie `True` si les deux insertions réussissent, et `False` si l'une échoue ou si les paramètres ne respectent pas les conditions vérifiées par la fonction.

Question 9.

Sujet p. 3, 4, 5

> Préciser les conditions que doivent vérifier les paramètres transmis à cette fonction.

`liste_valeurs` doit contenir exactement six valeurs dans l'ordre `marque`, `modele`, `kilometrage`, `nombre_places`, `type`, `carburant`. Les champs textuels doivent être des chaînes ; le kilométrage doit être un entier non négatif et le nombre de places un entier strictement positif, conformément aux données de la table. L'identifiant de voiture n'est pas fourni puisqu'il est auto-incrémenté. `id_agence` doit être un entier correspondant à une agence existante afin que la clé étrangère puisse être respectée.

> **RÉPONSE À RETENIR**
>
> Les paramètres doivent respecter le nombre, l'ordre, les types et les contraintes des attributs de la voiture ; `id_agence` doit désigner une agence existante.

Exercice 2

6 points

Réseaux, protocoles de routage et graphes

A

Question 1.

Sujet p. 6, 7

> En utilisant les adresses IP des différentes interfaces et des ordinateurs portables, en déduire une adresse possible pour le portable P2.

L1 est relié à R1, dont l'interface sur ce réseau a l'adresse 192.168.1.1/24. Le préfixe /24 fixe les trois premiers octets : le réseau est donc 192.168.1.0/24. P2 doit avoir une adresse de ce réseau, différente de celles déjà attribuées et des adresses réservées au réseau et à la diffusion.

> **RÉPONSE À RETENIR**
>
> Une adresse possible pour P2 est **192.168.1.11**. Elle appartient au réseau 192.168.1.0/24 et est différente des adresses de R1 et de P1.

Question 2.

Sujet p. 6, 7, 8

> Donner l'adresse du réseau local L2 ainsi que le nombre d'adresses possibles pour les ordinateurs portables P3 et P4.

L'interface de R9 raccordée à L2 est 192.168.18.1/24. Avec le préfixe /24, l'adresse réseau est 192.168.18.0 et l'adresse de diffusion est 192.168.18.255. Les 8 bits réservés à la partie hôte permettent donc $2^8 - 2 = 254$ adresses utilisables pour des machines. L'adresse 192.168.18.1 étant déjà attribuée à R9, il reste 253 adresses attribuables aux portables P3 et P4.

> **RÉPONSE À RETENIR**
>
> **Réseau L2 :** 192.168.18.0/24. Il y a 254 adresses utilisables pour les machines au total, dont une est celle de R9 ; **253 adresses restent donc disponibles pour P3 et P4**, de 192.168.18.2 à 192.168.18.254.

B

Question 3.

Sujet p. 8

> Donner l'implémentation Python des listes d'adjacence du graphe G à l'aide d'un dictionnaire dont les clés sont les sommets et les valeurs la liste des sommets adjacents du sommet clé. Nommer G ce dictionnaire. Chaque couple clé/valeur peut être écrit sur une nouvelle ligne.

On relève les voisins de chaque routeur dans la figure 2. Le graphe est non orienté : si Rj est voisin de Ri, alors Ri est aussi voisin de Rj. Les voisins sont listés ici dans l'ordre croissant des numéros.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
G = {
    'R1': ['R2', 'R3', 'R4', 'R6'],
    'R2': ['R1', 'R3', 'R5'],
    'R3': ['R1', 'R2', 'R5', 'R6'],
    'R4': ['R1', 'R6'],
    'R5': ['R2', 'R3', 'R6', 'R7'],
    'R6': ['R1', 'R3', 'R4', 'R5', 'R7', 'R8'],
    'R7': ['R5', 'R6', 'R8', 'R9'],
    'R8': ['R6', 'R7', 'R9'],
    'R9': ['R7', 'R8']
}
```

Question 4.

Sujet p. 8

En supposant que le protocole de routage RIP est utilisé, recopier et compléter, en ajoutant autant de lignes que nécessaire, la table de routage simplifiée du routeur R1. La table comporte les colonnes Destination, Suivant et Nombre de sauts ; la ligne R2 indique R2 comme suivant et 1 saut, et une ligne R3 est amorcée.

Avec RIP, le nombre de sauts correspond au nombre de liaisons entre routeurs. Depuis R1, les voisins directs R2, R3, R4 et R6 sont à 1 saut. R5, R7 et R8 sont à 2 sauts ; R9 est à 3 sauts. Pour R5, plusieurs suivants permettent d'obtenir le même nombre minimal de sauts : on choisit ici R2.

RÉPONSE À RETENIR

Destination	Suivant	Nombre de sauts
R2	R2	1
R3	R3	1
R4	R4	1
R5	R2	2
R6	R6	1
R7	R6	2
R8	R6	2
R9	R6	3

Pour R5, R3 peut également être choisi comme suivant : le trajet R1–R3–R5 comporte aussi 2 sauts.

Question 5.

Sujet p. 6, 8

L'ordinateur P1 envoie un paquet de données à l'ordinateur P3. Donner l'un des chemins empruntés par le paquet ainsi que le nombre de sauts.

P1 est sur le réseau local relié à R1 et P3 sur celui relié à R9. Un chemin entre ces routeurs est R1–R6–R8–R9 : il comporte trois liaisons entre routeurs.

RÉPONSE À RETENIR

Un chemin possible est **P1 → R1 → R6 → R8 → R9 → P3**. Il comporte **3 sauts entre routeurs**.

Question 6.

Sujet p. 8, 9

Donner l'écriture en Python de la matrice d'adjacence M du graphe de la figure 2 sous la forme d'une liste de listes. Les sommets sont rangés par ordre croissant des numéros des routeurs (R1, R2, …, R9).

La matrice d'adjacence contient 1 lorsque les deux routeurs sont reliés et 0 sinon. Les lignes et les colonnes suivent l'ordre R1, R2, …, R9. Le graphe étant non orienté, la matrice est symétrique et sa diagonale est nulle.

RÉPONSE À RETENIR

```
M = [
    [0, 1, 1, 1, 0, 1, 0, 0, 0],
    [1, 0, 1, 0, 1, 0, 0, 0, 0],
    [1, 1, 0, 0, 1, 1, 0, 0, 0],
    [1, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 1, 1, 0, 0, 1, 1, 0, 0],
    [1, 0, 1, 1, 1, 0, 1, 1, 0],
    [0, 0, 0, 0, 1, 1, 0, 1, 1],
    [0, 0, 0, 0, 0, 1, 1, 0, 1],
    [0, 0, 0, 0, 0, 0, 1, 1, 0]
]
```

Question 7.

Sujet p. 9

Recopier et compléter les lignes 3, 5 et 6 de la fonction degre. Elle prend en paramètre la matrice d'adjacence d'un graphe donné sous forme d'une liste de listes et renvoie la liste des degrés de tous les sommets du graphe, rangés dans le même ordre que les sommets de la matrice d'adjacence.

Pour chaque ligne, on additionne les valeurs de la ligne : chacune des valeurs 1 correspond à une arête incidente au sommet considéré. Les lignes 3, 5 et 6 complétées sont :

RÉPONSE À RETENIR

```
def degre(MATRICE):
    d = []
    for ligne in MATRICE:
        cpt = 0
        for valeur in ligne:
            cpt = cpt + valeur
        d.append(cpt)
    return d
```

Question 8.

Sujet p. 8, 9

> Donner la liste renvoyée par degre(M).

Le degré d'un routeur est la somme des valeurs de sa ligne dans la matrice. Par exemple, la ligne de R1 contient quatre 1 et celle de R6 en contient six.

> **RÉPONSE À RETENIR**
>
> Dans l'ordre R1 à R9, **degre(M) renvoie [4, 3, 4, 2, 4, 6, 4, 3, 2]**.

Question 9.

Sujet p. 8, 9

> En utilisant le résultat de la question précédente et en admettant que le graphe est connexe, indiquer si le robot peut parcourir l'ensemble du réseau en suivant les fibres optiques et en empruntant chaque fibre optique une et une seule fois. Une chaîne eulérienne est un chemin qui passe une et une seule fois par toutes les arêtes ; un graphe connexe en admet une si et seulement s'il possède au plus deux sommets de degré impair.

Les degrés impairs de la liste précédente sont ceux de R2 et de R8 : ils valent tous deux 3. Le graphe étant supposé connexe, il possède deux sommets de degré impair, donc au plus deux. D'après le critère donné dans le sujet, il admet une chaîne eulérienne.

> **RÉPONSE À RETENIR**
>
> **Oui.** Le robot peut parcourir chaque fibre une et une seule fois. Une chaîne eulérienne peut commencer en R2 et se terminer en R8, ou commencer en R8 et se terminer en R2.

C

Question 10.

Sujet p. 6, 10

> Déterminer la route qui sera empruntée par le paquet pour aller de l'ordinateur P1 à l'ordinateur P3 avec le protocole OSPF. Préciser le coût de ce trajet. Le poids de chaque arête représente la bande passante en mégabits par seconde (Mb/s) de la liaison et le coût d'une liaison est $C = 10^8 / BP$, où BP est la bande passante en bits par seconde.

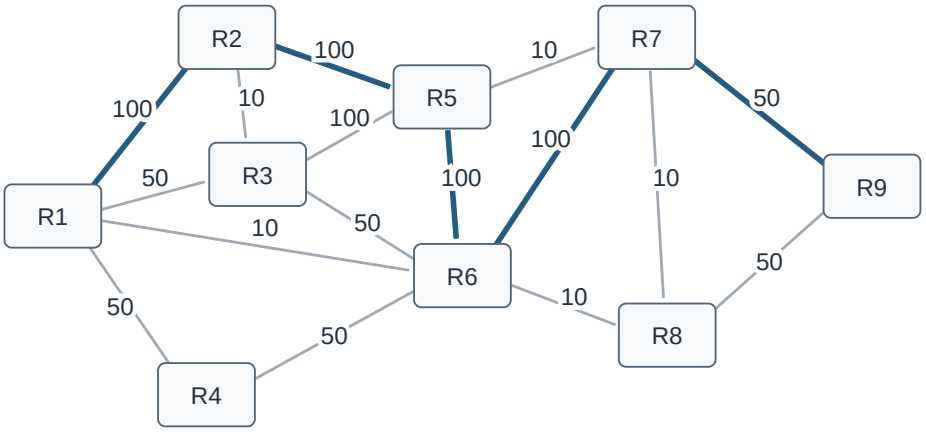
La bande passante de la figure est en Mb/s, alors que la formule demande des bits par seconde. Une liaison de 100 Mb/s correspond à 100 000 000 bits/s et a donc un coût de 1 ; une liaison de 50 Mb/s a un coût de 2 ; une liaison de 10 Mb/s a un coût de 10.

On peut appliquer la recherche du plus court chemin en additionnant ces coûts. Depuis R1, les coûts provisoires sont : R2 = 1, R3 = 2, R4 = 2 et R6 = 10. En passant par R2, on atteint R5 au coût 2. En passant par R3, on atteint R6 au coût 4. En passant par R5, le coût pour R6 devient 3. Depuis R6, on atteint R7 au coût 4. Depuis R7, on atteint R9 au coût 6. Cette progression donne le plus petit coût de R1 à R9 ; les prédécesseurs correspondants donnent R1–R2–R5–R6–R7–R9.

Le coût total est bien 1 + 1 + 1 + 1 + 2 = 6. Les liaisons entre les ordinateurs et leurs routeurs ne sont pas des arêtes du graphe pondéré de la figure 3 ; le coût demandé est celui du trajet entre les routeurs.

FormaV - document d'entraînement non officiel

Graphe pondéré de la figure 3 — bandes passantes en Mb/s



La route mise en évidence est R1–R2–R5–R6–R7–R9 ; les étiquettes reprennent les bandes passantes de la figure, en Mb/s.

> **RÉPONSE À RETENIR**
>
> La route OSPF est **P1 → R1 → R2 → R5 → R6 → R7 → R9 → P3**. Son coût entre les routeurs est **6**.

Exercice 3

8 points

Bases de numération, pile et programmation orientée objet

Question 1.

Sujet p. 11, 12, 16

> Compléter le tableau donné en annexe à rendre avec la copie, qui présente les étages 3, 2 et 1 de l'écriture Maya et leur valeur dans la conversion. La ligne de l'étage 3 est déjà renseignée.

Le chiffre Maya de l'étage 3 est déjà donné : un trait vaut 5 et un point vaut 1, donc sa valeur est 8. Pour les étages suivants, on lit les symboles de l'annexe : l'étage 2 comporte deux traits et un point ; l'étage 1 comporte trois traits. Le poids de l'étage 2 est 20^1 et celui de l'étage 1 est 20^0.

Étage	Écriture Maya	Valeur du « chiffre » de l'étage	Valeur dans la conversion
3	3 points et 1 trait	$1 \times 5 + 3 \times 1 = 8$	$8 \times 20^2 = 3200$
2	1 point et 2 traits	$2 \times 5 + 1 \times 1 = 11$	$11 \times 20^1 = 220$
1	3 traits	$3 \times 5 + 0 \times 1 = 15$	$15 \times 20^0 = 15$

RÉPONSE À RETENIR

À compléter dans l'annexe : étage 2, valeur du chiffre 11 et valeur convertie 220 ; étage 1, valeur du chiffre 15 et valeur convertie 15.

Question 2.

Sujet p. 11, 12

> Justifier que l'écriture Maya représentée dans la figure 2 est celle de l'entier 3435.

Dans la figure 2, l'étage supérieur comporte un trait et trois points : sa valeur est $5 + 3 = 8$. L'étage du milieu comporte deux traits et un point : sa valeur est $2 \times 5 + 1 = 11$. L'étage inférieur comporte trois traits : sa valeur est $3 \times 5 = 15$. Ces étages ont respectivement pour poids 20^2, 20^1 et 20^0.

La valeur représentée est donc $8 \times 20^2 + 11 \times 20^1 + 15 \times 20^0 = 8 \times 400 + 11 \times 20 + 15 = 3200 + 220 + 15 = 3435$.

RÉPONSE À RETENIR

L'écriture Maya de la figure 2 représente bien l'entier 3435.

Question 3.

Sujet p. 12, 13

> Écrire une suite d'instructions permettant de créer une instance nommée M de la classe Maya qui modélise le nombre entier 3435.

La liste modélise une pile dont le sommet est en fin de liste. Il faut donc empiler d'abord le chiffre de poids 20^0 (l'étage inférieur), puis celui de poids 20^1, puis celui de poids 20^2.

FormaV - document d'entraînement non officiel

```
M = Maya()
M.ajouter([0, 0, 3])
M.ajouter([0, 1, 2])
M.ajouter([0, 3, 1])
```

Après les trois ajouts, `M.nombre` vaut `[[0, 0, 3], [0, 1, 2], [0, 3, 1]]`, la modélisation de 3435 donnée dans le sujet.

> **RÉPONSE À RETENIR**
>
> La suite d'instructions ci-dessus crée l'objet `M` modélisant 3435.

Question 4.

Sujet p. 13

> Écrire la méthode nbEtages de la classe Maya. Celle-ci renvoie le nombre de « chiffres » utilisés pour écrire le nombre correspondant en écriture Maya.

Chaque élément de `self.nombre` représente un étage ; le nombre d'étages est donc le nombre d'éléments de cette liste. La méthode ne modifie pas la pile.

```
def nbEtages(self):
    return len(self.nombre)
```

Par exemple, pour l'objet de la question 3, la liste contient trois chiffres et `M.nbEtages()` renvoie 3.

> **RÉPONSE À RETENIR**
>
> La méthode renvoie `len(self.nombre)`.

Question 5.

Sujet p. 11, 14

> Écrire une fonction valeurChiffre ayant pour paramètre une liste L. Celle-ci renvoie la valeur de l'entier associé à la liste L = [c, p, t], où c (de valeur 0 ou 1) indique la présence d'une coquille, p est le nombre de points et t le nombre de traits composant un « chiffre » Maya. Exemples : valeurChiffre([0, 2, 3]) renvoie 17 et valeurChiffre([1, 0, 0]) renvoie 0.

Un point vaut 1 et un trait vaut 5. Si `c` vaut 1, le chiffre est la coquille, qui représente zéro. Sinon, la valeur du chiffre est `p + 5 * t`.

```
def valeurChiffre(L):
    c, p, t = L
    if c == 1:
        return 0
    return p + 5 * t
```

Pour `[0, 2, 3]`, cela donne 2 + 5 × 3 = 17. Pour `[1, 0, 0]`, la coquille vaut 0.

> **RÉPONSE À RETENIR**
>
> `valeurChiffre([0, 2, 3])` renvoie 17 et `valeurChiffre([1, 0, 0])` renvoie 0.

Question 6.

Sujet p. 13, 14

> Recopier et compléter les lignes 2, 4, 5 et 7 de la méthode MayaToDec de la classe Maya. Cette méthode renvoie la valeur de l'entier associé à l'objet Maya. On pourra utiliser les méthodes estVide, nbEtages et retirer.

Les chiffres sont empilés du poids le plus faible au poids le plus fort. Le premier chiffre retiré est donc celui du sommet, de poids $20^{\text{nbEtages}() - 1}$. Après chaque retrait, le coefficient doit être divisé par 20. La division entière conserve le type entier du résultat. Le cas de la pile vide est traité pour ne pas calculer un exposant négatif inutile ; la boucle renvoie alors 0.

```
def MayaToDec(self):
    n = self.nbEtages()
    coeff = 20 ** (n - 1) if n > 0 else 1
    ch_Dec = 0
    while not self.estVide():
        ch_Maya = self.retirer()
        ch_Dec = ch_Dec + valeurChiffre(ch_Maya) * coeff
        coeff = coeff // 20
    return ch_Dec
```

Trace pour 3435 : les coefficients successifs sont 400, 20 et 1. Les chiffres retirés valent 8, 11 et 15 ; leurs contributions sont respectivement 3200, 220 et 15, soit un total de 3435. Comme la méthode utilise `retirer`, elle vide la pile de l'objet.

RÉPONSE À RETENIR

La méthode renvoie l'entier représenté ; pour la modélisation de 3435, elle renvoie 3435. Pour une pile vide, elle renvoie 0.

Question 7.

Sujet p. 11, 14, 15

> Écrire la fonction decompChiffre, qui prend en paramètre un entier n compris entre 0 et 19 et renvoie la liste [c, p, t] correspondant au « chiffre » Maya associé. c vaut 0 ou 1 et indique la présence ou non d'une coquille, p est le nombre de points et t le nombre de traits. Exemples : decompChiffre(17) renvoie [0, 2, 3] et decompChiffre(0) renvoie [1, 0, 0].

Pour un chiffre Maya non nul, la division euclidienne de `n` par 5 fournit le nombre de traits `n // 5` et le nombre de points `n % 5`. Le chiffre 0 est représenté par la coquille.

```
def decompChiffre(n):
    if n == 0:
        return [1, 0, 0]
    p = n % 5
    t = n // 5
    return [0, p, t]
```

Par exemple, 17 = 3 × 5 + 2 donne deux points et trois traits, soit `[0, 2, 3]`.

RÉPONSE À RETENIR

`decompChiffre(17)` renvoie `[0, 2, 3]` et `decompChiffre(0)` renvoie `[1, 0, 0]`.

Question 8.

Sujet p. 12, 13, 14, 15

> Écrire la fonction DecToMaya, qui prend en paramètre un entier n et renvoie la modélisation Maya d'un objet M de la classe Maya correspondant. La fonction DecToVige est supposée déjà écrite : elle renvoie la décomposition en base 20 sous la forme $[a_0, a_1, \ldots, a_p]$, telle que $n = a_0 \times 20^0 + a_1 \times 20^1 + \ldots + a_p \times 20^p$. Exemples : DecToMaya(3435).nombre renvoie [[0, 0, 3], [0, 1, 2], [0, 3, 1]] et DecToMaya(407).nombre renvoie [[0, 2, 1], [1, 0, 0], [0, 1, 0]].

`DecToVige(n)` fournit les chiffres de base 20 du poids le plus faible au poids le plus fort. On convertit chacun en chiffre Maya et on l'empile dans cet ordre, puisque le sommet de la pile est en fin de liste.

```
def DecToMaya(n):
    M = Maya()
    chiffres = DecToVige(n)
    for chiffre in chiffres:
        M.ajouter(decompChiffre(chiffre))
    return M
```

Pour 3435, la décomposition fournie est `[15, 11, 8]`, qui devient successivement `[0, 0, 3]`, `[0, 1, 2]` et `[0, 3, 1]`. Pour 407, `[7, 0, 1]` devient `[0, 2, 1]`, `[1, 0, 0]` et `[0, 1, 0]`.

RÉPONSE À RETENIR

`DecToMaya(3435).nombre` vaut `[[0, 0, 3], [0, 1, 2], [0, 3, 1]]` et `DecToMaya(407).nombre` vaut `[[0, 2, 1], [1, 0, 0], [0, 1, 0]]`.

Question 9.

Sujet p. 13, 15

> Écrire la méthode multiplie de la classe Maya, qui renvoie le résultat de la multiplication par 20 d'un nombre en modélisation Maya.

Multiplier par 20 décale chaque chiffre d'un étage vers le haut. On obtient donc un chiffre zéro au poids 20^0, puis les chiffres initiaux dans leur ordre. La méthode crée un nouvel objet et ne modifie pas celui de départ.

```
def multiplie(self):
    resultat = Maya()
    resultat.ajouter([1, 0, 0])
    for chiffre in self.nombre:
        resultat.ajouter(chiffre[:])
    return resultat
```

Avec l'exemple du sujet, la pile initiale est `[[0, 0, 3], [0, 1, 2]]`. Après ajout de la coquille puis copie des deux chiffres, la pile obtenue est `[[1, 0, 0], [0, 0, 3], [0, 1, 2]]`.

RÉPONSE À RETENIR

La méthode renvoie une nouvelle modélisation Maya représentant 20 fois le nombre initial.

Question 10. — premier appel

Sujet p. 15

> Donner le résultat renvoyé par l'appel mystere([0, 1, 1], [0, 3, 1], 0).

La fonction imprimée dans le sujet contient à la ligne 13 `p + t = 0`. En Python, cette expression n'est pas une comparaison valide : telle qu'imprimée, la fonction produit une erreur de syntaxe et l'appel ne renvoie donc aucun résultat. Pour donner le résultat vraisemblablement attendu, on interprète ce signe comme une comparaison `p + t == 0`.

Étape	Calcul	Valeur obtenue
Points	(1 + 3 + 0) % 5 = 4 ; la somme vaut moins de 5	p = 4, retenue = 0
Traits	(1 + 1 + 0) % 4 = 2 ; la somme vaut moins de 4	t = 2, retenue finale = 0
Coquille	Les chiffres d'entrée n'ont pas de coquille et p + t = 6	c = 0

RÉPONSE À RETENIR

Au sens strict, l'appel ne renvoie rien car le code imprimé est syntaxiquement invalide. Si l'on corrige la comparaison en `p + t == 0`, le résultat attendu est `([0, 4, 2], 0)`.

Question 10. — second appel

Sujet p. 15

> Donner le résultat renvoyé par l'appel mystere([0, 1, 1], [0, 4, 2], 0).

La fonction imprimée dans le sujet contient à la ligne 13 `p + t = 0`. En Python, cette expression n'est pas une comparaison valide : telle qu'imprimée, la fonction produit une erreur de syntaxe et l'appel ne renvoie donc aucun résultat. Pour donner le résultat vraisemblablement attendu, on interprète ce signe comme une comparaison `p + t == 0`.

Étape	Calcul	Valeur obtenue
Points	(1 + 4 + 0) % 5 = 0 ; la somme vaut au moins 5	p = 0, retenue = 1
Traits	(1 + 2 + 1) % 4 = 0 ; la somme vaut au moins 4	t = 0, retenue finale = 1
Coquille	p + t = 0 et la retenue finale vaut 1	c = 1

RÉPONSE À RETENIR

Au sens strict, l'appel ne renvoie rien car le code imprimé est syntaxiquement invalide. Si l'on corrige la comparaison en `p + t == 0`, le résultat attendu est `([1, 0, 0], 1)`.

Question 11.

Sujet p. 13, 15

> Écrire une méthode somme de la classe Maya permettant d'ajouter à l'instance courante un autre nombre maya2 de même taille en modélisation Maya.

Les chiffres sont rangés du poids le plus faible au poids le plus fort. On additionne donc les chiffres de même indice, en ajoutant la retenue de l'étage précédent. La valeur obtenue à chaque étage est décomposée en un chiffre compris entre 0 et 19 et une retenue. La méthode ci-dessous suppose, comme le demande l'énoncé, que les deux objets ont le même nombre d'étages ; elle renvoie `None` si cette condition n'est pas respectée. Si une retenue subsiste après le dernier étage, elle est ajoutée comme nouvel étage.

```
def somme(self, maya2):
    if self.nbEtages() != maya2.nbEtages():
        return None

    resultat = Maya()
    retenue = 0
    for i in range(self.nbEtages()):
        total = (valeurChiffre(self.nombre[i])
                 + valeurChiffre(maya2.nombre[i])
                 + retenue)
        resultat.ajouter(decompChiffre(total % 20))
        retenue = total // 20

    if retenue > 0:
        resultat.ajouter(decompChiffre(retenue))

    self.nombre = resultat.nombre
    return self
```

Exemple de trace avec les modélisations de 3435 et 407, qui ont toutes deux trois étages : à l'étage de poids 1, 15 + 7 = 22 donne le chiffre 2 et une retenue 1 ; à l'étage de poids 20, 11 + 0 + 1 = 12 donne le chiffre 12 et aucune retenue ; à l'étage de poids 400, 8 + 1 = 9. Les chiffres obtenus valent donc 2, 12 et 9, soit 2 + 12 × 20 + 9 × 400 = 3842.

> **RÉPONSE À RETENIR**
>
> La méthode modifie l'instance courante et la renvoie. Pour 3435 + 407, elle produit `[[0, 2, 0], [0, 2, 2], [0, 4, 1]]`, qui représente 3842.

Sujet : https://www.sujetdebac.fr/annales-pdf/2024/spe-numerique-informatique-2024-metropole-1-remplacement-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.