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.

Numérique et sciences informatiques

Session 2025 • Polynésie • Jour 1

Code sujet **25-NSIJ1PO1** 3 heures 30

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Protocoles réseaux, algorithmique et programmation orientée objet
10 réponses • sujet p. 2–7

Exercice 2 — Programmation Python et récursivité : le baguenaudier
10 réponses • sujet p. 8–11

Exercice 3 — Programmation Python, bases de données relationnelles, SQL, systèmes d’exploitation et sécurisation des communications
18 réponses • sujet p. 12–19

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

Protocoles réseaux, algorithmique et programmation orientée objet

A

Question 1.

Sujet p. 2

> Recopier et compléter la table de routage de R1 sachant que le protocole de routage RIP est utilisé. La destination est le routeur à atteindre, le prochain saut est le premier routeur traversé et la distance est le nombre de routeurs traversés. Le tableau comporte les destinations R2 à R6.

Avec RIP, on choisit un chemin qui traverse le moins de routeurs possible. Le tableau donne la convention de distance : la liaison directe vers R2 a une distance 0, et la route de R1 à R3 via R2 a une distance 1. Les routeurs directement reliés à R1 ont donc une distance 0. Pour R6, le chemin le plus court est R1-R5-R6 : le prochain saut est R5 et la distance est 1.

Destination	Prochain saut	Distance
R2	R2	0
R3	R2	1
R4	R4	0
R5	R5	0
R6	R5	1

RÉPONSE À RETENIR

Table de routage RIP de R1 : R2 via R2, distance 0 ; R3 via R2, distance 1 ; R4 via R4, distance 0 ; R5 via R5, distance 0 ; R6 via R5, distance 1.

Question 2.

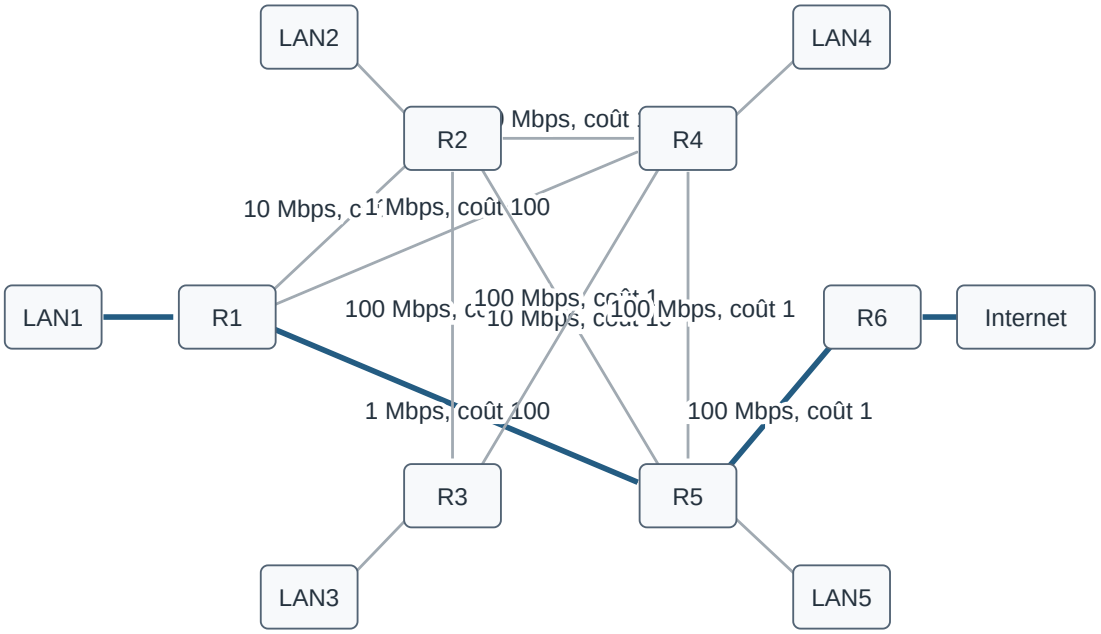
Sujet p. 2, 3

> Un utilisateur du réseau local LAN1 interroge, via son navigateur web, un serveur d'Internet. Détailler, suivant le protocole RIP, la route suivie dans le maillage par la requête à destination de ce serveur.

RIP privilégie le chemin comportant le moins de routeurs intermédiaires, sans tenir compte du débit des liaisons. De R1 à R6, le chemin le plus court est R1-R5-R6. La requête part donc de LAN1, rejoint R1, traverse R5 puis R6, et atteint Internet.

FormaV - document d'entraînement non officiel

Réseau et route RIP de LAN1 vers Internet



Les débits et les coûts des liaisons sont indiqués ; le trajet RIP est mis en évidence.

> **RÉPONSE À RETENIR**
>
> Route suivie : LAN1 → R1 → R5 → R6 → Internet.

Question 3.

Sujet p. 2, 3

> Recopier et compléter la table de routage de R1 sachant que le protocole OSPF est utilisé. La distance est la somme totale des coûts des liaisons physiques empruntées pour atteindre la destination. Le coût d'une liaison est $10^2/d$, où d est son débit en Mbps ; les coûts des liaisons depuis R1 vers R2, R4 et R5 sont respectivement 10, 100 et 100. Le tableau comporte les destinations R2 à R6.

Pour OSPF, on additionne les coûts des liaisons, avec coût = 10^2 / débit. Ainsi, une liaison de 100 Mbps coûte 1, une liaison de 10 Mbps coûte 10 et une liaison de 1 Mbps coûte 100. On compare les chemins possibles depuis R1 :

- R2 : liaison directe, coût 10.
- R3 : R1-R2-R3, coût 10 + 1 = 11.
- R4 : R1-R2-R3-R4, coût 10 + 1 + 1 = 12, inférieur au coût 20 du chemin R1-R2-R4 et au coût 100 de la liaison directe.
- R5 : R1-R2-R3-R4-R5, coût 10 + 1 + 1 + 1 = 13.
- R6 : R1-R2-R3-R4-R5-R6, coût 10 + 1 + 1 + 1 + 1 = 14.

Les chemins optimaux vers R3, R4, R5 et R6 commencent tous par R1-R2 : leur prochain saut est donc R2.

Destination	Prochain saut	Distance
R2	R2	10
R3	R2	11
R4	R2	12
R5	R2	13
R6	R2	14

RÉPONSE À RETENIR

Table de routage OSPF de R1 : R2 via R2, distance 10 ; R3 via R2, distance 11 ; R4 via R2, distance 12 ; R5 via R2, distance 13 ; R6 via R2, distance 14.

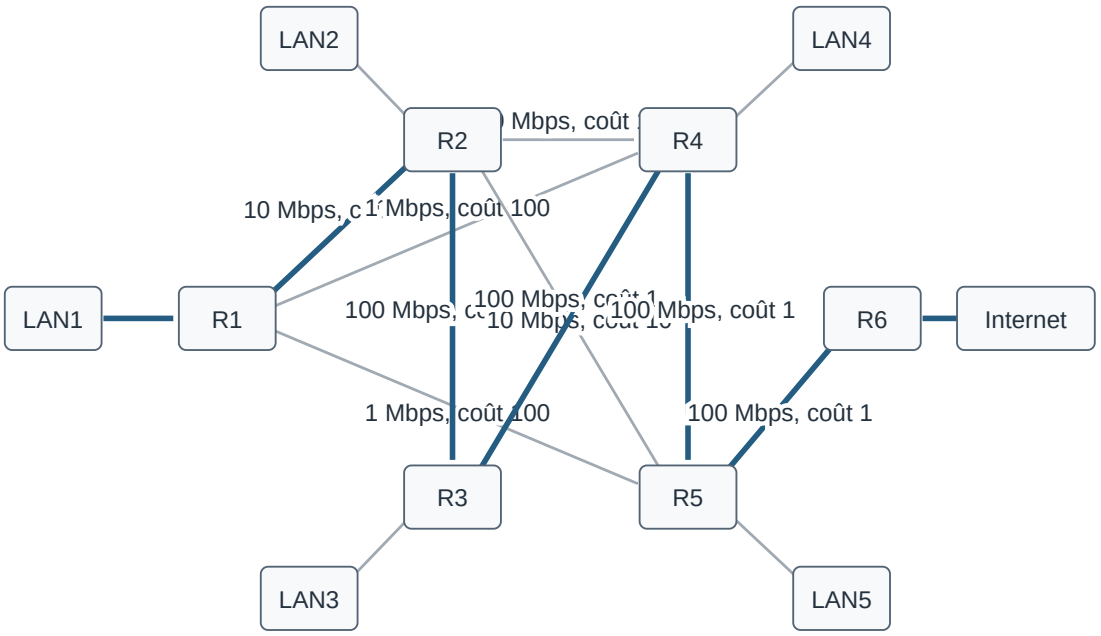
Question 4.

Sujet p. 2, 3

Un utilisateur du réseau local LAN1 interroge, via son navigateur web, un serveur d'Internet. Donner, selon le protocole OSPF, la route suivie dans le maillage par la requête à destination de ce serveur.

Le coût du chemin de R1 à R6 passant par R2, R3, R4 et R5 est 10 + 1 + 1 + 1 + 1 = 14. OSPF choisit ce chemin, dont le coût est inférieur à celui des autres itinéraires possibles. La liaison de R6 à Internet complète la route ; aucun coût n'est donné pour cette liaison.

Réseau et route OSPF de LAN1 vers Internet



La route OSPF de coût 14 jusqu'à R6 est mise en évidence.

RÉPONSE À RETENIR

Route selon OSPF : LAN1 → R1 → R2 → R3 → R4 → R5 → R6 → Internet. La distance dans le maillage jusqu'à R6 est 14.

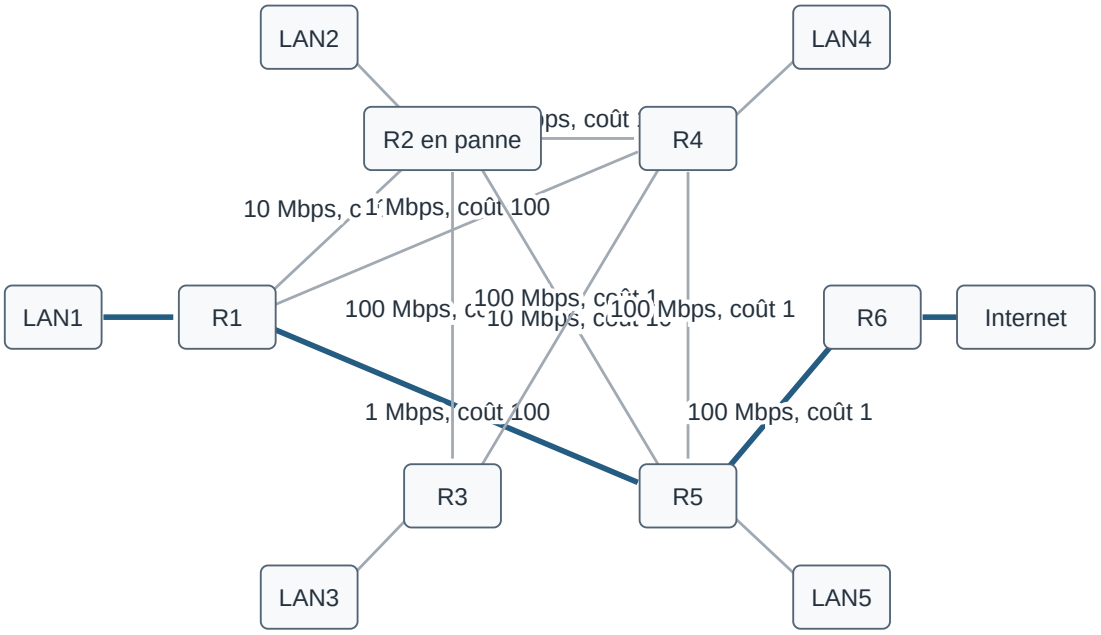
Question 5.

Sujet p. 2, 3

> Le protocole de routage OSPF est toujours utilisé et le routeur R2 tombe en panne. Donner la nouvelle route suivie dans le maillage par une requête partant du réseau LAN1 à destination d'Internet et en donner la nouvelle valeur de la distance.

La panne de R2 exclut les itinéraires qui passent par ce routeur. Le chemin R1-R5-R6 coûte 100 + 1 = 101. L'autre itinéraire possible par R4, R1-R4-R5-R6, coûte 100 + 1 + 1 = 102. OSPF choisit donc le premier.

Route OSPF après la panne de R2



R2 est en panne et n'est pas emprunté ; la route de coût 101 est mise en évidence.

RÉPONSE À RETENIR

Nouvelle route : LAN1 → R1 → R5 → R6 → Internet. La nouvelle distance dans le maillage jusqu'à R6 est 101.

B

Question 6.

Sujet p. 4

> Recopier et compléter le tableau correspondant à une machine d'adresse 192.168.1.100 et de masque 255.192.0.0. Ce tableau détermine l'adresse du réseau, en binaire puis en décimale pointée.

On effectue un ET bit à bit entre chaque octet de l'adresse machine et l'octet correspondant du masque. Les quatre octets de la machine sont 11000000, 10101000, 00000001 et 01100100. Par exemple, 10101000 ET 11000000 = 10000000.

FormaV - document d'entraînement non officiel

	1er octet	2e octet	3e octet	4e octet
Machine (binaire)	11000000	10101000	00000001	01100100
Masque (binaire)	11111111	11000000	00000000	00000000
Réseau (binaire)	11000000	10000000	00000000	00000000
Réseau (décimale pointée)	192	128	0	0

RÉPONSE À RETENIR

L'adresse du réseau est 11000000.10000000.00000000.00000000 en binaire, soit 192.128.0.0 en décimale pointée.

Question 7.

Sujet p. 4, 5

Recopier et compléter le tableau pour déterminer l'adresse de broadcast du réseau suivant, en binaire puis en décimale pointée : adresse réseau binaire 11000000 10000000 00000000 00000000 et masque binaire 11111111 11000000 00000000 00000000.

On complète chaque octet du masque en échangeant les 0 et les 1, puis on effectue le OU bit à bit avec l'adresse réseau. Au deuxième octet, 10000000 OU 00111111 = 10111111. Les deux derniers octets du broadcast valent 11111111.

	1er octet	2e octet	3e octet	4e octet
Réseau (binaire)	11000000	10000000	00000000	00000000
Masque (binaire)	11111111	11000000	00000000	00000000
Complément du masque (binaire)	00000000	00111111	11111111	11111111
Broadcast (binaire)	11000000	10111111	11111111	11111111
Broadcast (décimale pointée)	192	191	255	255

RÉPONSE À RETENIR

L'adresse de broadcast est 11000000.10111111.11111111.11111111 en binaire, soit 192.191.255.255 en décimale pointée.

Question 8.

Sujet p. 5

Une machine du réseau LAN1 a reçu du serveur DHCP l'adresse 172.16.1.100, avec le masque 255.255.0.0. Sa passerelle par défaut a pour adresse 172.16.255.254. En déduire l'adresse du réseau LAN1 en décimale pointée, l'adresse de broadcast en décimale pointée et le nombre total d'adresses pouvant être distribuées par le serveur, sans tenir compte d'éventuelles restrictions posées par l'administrateur du réseau.

Le masque 255.255.0.0 correspond à 16 bits de réseau, soit un réseau /16. L'application du masque à 172.16.1.100 donne le réseau 172.16.0.0. En mettant à 1 les 16 bits de la partie hôte, on obtient le broadcast 172.16.255.255.

Il y a 16 bits pour la partie hôte, donc 2^{16} = 65 536 adresses au total. En excluant l'adresse réseau et l'adresse de broadcast, il reste 65 534 adresses d'hôte utilisables. Puisque la passerelle utilise 172.16.255.254, si l'on demande le nombre d'adresses pouvant être louées à d'autres machines, il en reste au plus 65 533.

> **RÉPONSE À RETENIR**
>
> Adresse du réseau : 172.16.0.0. Adresse de broadcast : 172.16.255.255. Le réseau comporte 65 534 adresses d'hôte utilisables, dont une est déjà celle de la passerelle ; il peut donc rester au plus 65 533 adresses à distribuer aux autres machines.

Question 9.

Sujet p. 6, 7

> Recopier et compléter les lignes 15 et 16 du code de la méthode masquer de la classe IPv4, afin que la méthode corresponde à son docstring. Elle détermine le préfixe masqué de l'adresse à partir d'un masque en décimale pointée ; l'exemple donné est IPv4('192.168.1.100').masquer('255.192.0.0'), qui renvoie '192.128.0.0'.

Pour chacun des quatre indices, on applique le ET bit à bit entre l'octet de l'adresse et celui du masque, puis on ajoute l'octet obtenu à `tmp`. Comme `join` assemble des chaînes, il faut convertir les entiers en chaînes.

```
tmp.append(ip[i] & crible[i])
return ".".join(str(i) for i in tmp)
```

Avec l'exemple, les octets obtenus sont 192 & 255 = 192, 168 & 192 = 128, 1 & 0 = 0 et 100 & 0 = 0. La méthode renvoie donc bien `192.128.0.0`.

> **RÉPONSE À RETENIR**
>
> Ligne 15 : `tmp.append(ip[i] & crible[i])`. Ligne 16 : `return ".".join(str(i) for i in tmp)`.

Question 10.

Sujet p. 7

> Recopier et compléter les lignes 19 et 20 du code de la méthode adresse_suivante de la classe IPv4, afin que la méthode corresponde à son docstring. Elle détermine l'adresse décimale pointée suivant immédiatement l'adresse courante, sous réserve de l'existence d'une adresse disponible. Les exemples portent sur les adresses 192.168.1.100 et 192.168.1.255.

Le calcul commence par l'octet le moins significatif, à l'indice 3. La valeur de l'octet est le reste de la division de `somme` par 256 et la retenue est le quotient. Les octets étant traités de droite à gauche, chaque résultat doit être ajouté au début de la liste pour rétablir l'ordre habituel.

```
valeur, retenue = somme % 256, somme // 256
liste_suivante = [str(valeur)] + liste_suivante
```

Pour 192.168.1.255, le dernier octet augmenté de 1 vaut 256 : on inscrit 0 et on transmet une retenue de 1. L'octet précédent vaut alors 1 + 1 = 2, sans retenue supplémentaire ; les deux premiers restent 192 et 168. On obtient 192.168.2.0. Pour 192.168.1.100, seul le dernier octet change et l'adresse suivante est 192.168.1.101.

RÉPONSE À RETENIR

Ligne 19 : `valeur, retenue = somme % 256, somme // 256`. Ligne 20 : `liste_suivante = [str(valeur)] + liste_suivante`.

Exercice 2

6 points

Programmation Python et récursivité : le baguenaudier

Question 1.

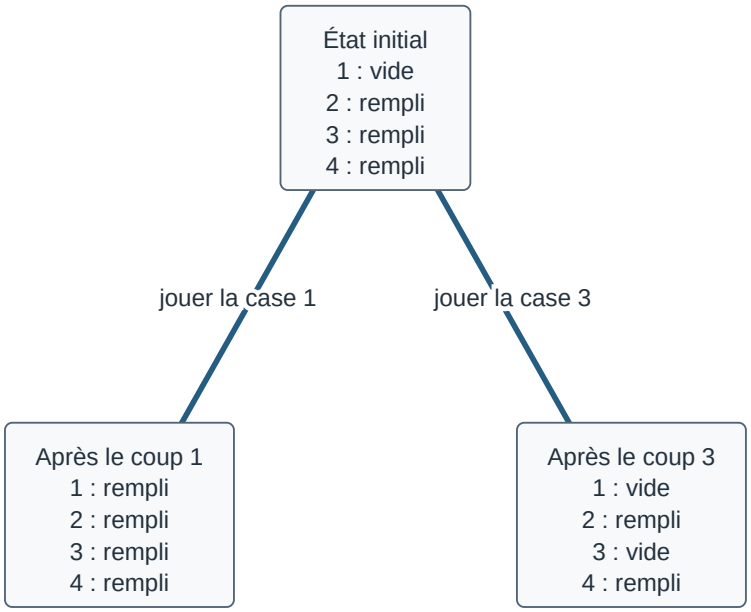
Sujet p. 8

> On considère un baguenaudier à 4 cases sur lequel les trois dernières cases sont remplies. Désigner les cases que l'on peut jouer et dessiner l'état du baguenaudier à la suite de chacun des coups possibles.

L'état initial est `[False, True, True, True]` : les cases 2, 3 et 4 sont remplies. La case 1 est toujours jouable. Comme le baguenaudier n'est ni vide ni entièrement rempli, on peut aussi jouer la case qui suit la première case remplie. La première case remplie est la case 2 : on peut donc jouer la case 3.

Jouer une case vide la remplit ; jouer une case remplie la vide. Attention : les numéros des cases commencent à 1, contrairement aux indices d'une liste Python.

Coups possibles pour le baguenaudier à 4 cases



Un pion est posé dans une case vide et retiré d'une case remplie.

RÉPONSE À RETENIR

Les cases jouables sont les cases **1 et 3**.

- Après avoir joué la case 1 : `[True, True, True, True]`.
- Après avoir joué la case 3 : `[False, True, False, True]`.

Question 2.

Sujet p. 9

> Écrire le code de la fonction initialiser, qui prend en paramètre un nombre entier n et renvoie une liste modélisant un baguenaudier vide de n cases.

Une liste de `n` cases vides se construit avec `[False] * n` : cette expression répète `n` fois la valeur `False`. On suppose que `n` est un entier positif ou nul.

FormaV - document d'entraînement non officiel

```
def initialiser(n):
    return [False] * n
```

Pour l'exemple de l'énoncé, `initialiser(3)` renvoie `[False, False, False]`.

RÉPONSE À RETENIR

```
def initialiser(n):
    return [False] * n
```

Question 3.

Sujet p. 9

> Recopier et compléter les lignes 3, 4 et 5 du code de la fonction victoire, qui renvoie un booléen indiquant si toutes les cases du baguenaudier sont remplies.

Une case vide est représentée par `False`. Dès que la boucle en rencontre une, la fonction renvoie donc `False`. Si la boucle se termine sans en trouver, toutes les cases sont remplies et la fonction renvoie `True`.

```
def victoire(tab):
    for etat_case in tab:
        if etat_case == False:
            return False
    return True
```

Trace : pour `[True, False, True]`, la boucle finit par rencontrer `False` et renvoie `False`. Pour `[True, True]`, elle ne rencontre aucune case vide et renvoie `True`. Pour une liste vide, le code renvoie également `True`, puisqu'aucune case vide n'est rencontrée.

RÉPONSE À RETENIR

Ligne 3 : `False` ; ligne 4 : `False` ; ligne 5 : `return True`.

Question 4.

Sujet p. 9

> Écrire le code de la fonction indice_premiere_case_occupee. Elle prend en paramètre un tableau tab modélisant l'état du baguenaudier et renvoie l'indice de la première case remplie, ou None si le baguenaudier est vide.

On parcourt le tableau dans l'ordre des indices. Au premier élément égal à `True`, on renvoie son indice. Si la boucle se termine sans trouver de case remplie, on renvoie `None`.

```
def indice_premiere_case_occupee(tab):
    for indice in range(len(tab)):
        if tab[indice] == True:
            return indice
    return None
```

Trace avec les exemples de l'énoncé : `[True, True, True]` renvoie `0` ; `[False, False, False, True, True]` renvoie `3` ; un tableau entièrement vide, tel que `[False, False]`, renvoie `None`.

RÉPONSE À RETENIR

```
def indice_premiere_case_occupee(tab):
    for indice in range(len(tab)):
        if tab[indice] == True:
            return indice
    return None
```

Question 5.

Sujet p. 8, 9

> Écrire le code de la fonction coup_valide. Elle prend en paramètres tab, une liste modélisant l'état d'un baguenaudier, et un entier case, indice de la case à jouer. Elle renvoie True si le coup respecte les règles et False sinon. L'indice de la première case est 0 ; vérifier également que l'indice est un entier positif inférieur à la longueur de tab.

Il faut d'abord vérifier que `case` est un entier correspondant à un indice valide : `0 <= case < len(tab)`. L'indice 0 représente la case 1, toujours jouable. Pour tout autre indice, le tableau doit être ni vide ni entièrement rempli, et l'indice demandé doit suivre celui de la première case remplie.

Le code utilise les fonctions définies précédemment. Le test `type(case) is not int` écarte également les booléens, bien que `bool` soit une sous-classe de `int` en Python.

```
def coup_valide(tab, case):
    if type(case) is not int or case < 0 or case >= len(tab):
        return False
    if case == 0:
        return True

    premier = indice_premiere_case_occupee(tab)
    if premier is None or victoire(tab):
        return False
    return case == premier + 1
```

Trace : avec `[False, False, True, True]`, la première case remplie a l'indice 2. Les indices jouables sont donc 0 et 3, soit les cases 1 et 4. Un indice négatif ou égal à `len(tab)` est rejeté avant les autres tests.

RÉPONSE À RETENIR

```
def coup_valide(tab, case):
    if type(case) is not int or case < 0 or case >= len(tab):
        return False
    if case == 0:
        return True

    premier = indice_premiere_case_occupee(tab)
    if premier is None or victoire(tab):
        return False
    return case == premier + 1
```

Question 6.

Sujet p. 8, 9, 10

> Écrire le code de la fonction changer_case. Elle prend en paramètres un tableau tab modélisant l'état du baguenaudier et un indice case. Si le coup désigné par case est valide, la fonction renvoie le tableau obtenu après avoir changé l'état de la case correspondante. Si le coup n'est pas valide, le baguenaudier passé en paramètre est renvoyé sans modification.

La fonction vérifie d'abord la validité du coup. Si le coup est valide, elle inverse l'état de la case : `False` devient `True`, et `True` devient `False`. Si le coup est invalide, elle ne modifie pas le tableau. Dans les deux cas, elle renvoie le tableau.

```
def changer_case(tab, case):
    if coup_valide(tab, case):
        tab[case] = not tab[case]
    return tab
```

Traces conformes aux exemples de l'énoncé : avec `[False, False, False]`, jouer l'indice 1 est invalide et laisse le tableau inchangé ; jouer l'indice 0 donne `[True, False, False]`. Avec `[False, True, False]`, jouer l'indice 2 donne `[False, True, True]`.

RÉPONSE À RETENIR

```
def changer_case(tab, case):
    if coup_valide(tab, case):
        tab[case] = not tab[case]
    return tab
```

Question 7.

Sujet p. 11

> Recopier et compléter la ligne 6 du code de la fonction vider pour qu'elle affiche le texte 'Vider case ' suivi de la valeur de l'entier n.

La ligne doit afficher le texte `Vider case ` , suivi de la valeur de l'entier `n`. On convertit donc `n` en chaîne avec `str(n)` avant la concaténation.

```
print('Vider case ' + str(n))
```

RÉPONSE À RETENIR

Ligne 6 : `print('Vider case ' + str(n))`.

Question 8.

Sujet p. 11

> En supposant que l'appel remplir(1) affiche ' Remplir case 1' et que l'appel remplir(0) ne fait rien, donner les affichages, dans la console, produits par l'exécution de vider(3).

Dans `vider(3)`, l'appel `vider(1)` affiche d'abord `Vider case 1`. La fonction affiche ensuite `Vider case 3`, puis appelle `remplir(1)`. Enfin, l'appel `vider(2)` commence par `vider(0)`, qui ne fait rien, affiche `Vider case 2`, appelle `remplir(0)`, qui ne fait rien, puis affiche `Vider case 1`. L'espace initial de l'affichage de `remplir(1)` est celui indiqué dans l'énoncé.

RÉPONSE À RETENIR

```
Vider case 1
Vider case 3
 Remplir case 1
Vider case 2
Vider case 1
```

Question 9.

Sujet p. 10, 11

> En vous inspirant de la fonction vider, de la résolution donnée à la question précédente pour n = 3 et des Figures 2 et 3, écrire le code de la fonction récursive remplir.

La fonction qui remplit le baguenaudier suit le parcours inverse de celui de `vider`. Pour remplir `n` cases, elle remplit d'abord les `n - 1` premières, vide les `n - 2` premières, remplit la case `n`, puis remplit les `n - 2` premières. Le cas `n == 1` est le cas de base ; pour `n == 0`, aucun affichage n'est produit.

```
def remplir(n):
    if n == 1:
        print('Remplir case 1')
    elif n > 1:
        remplir(n - 1)
        vider(n - 2)
        print('Remplir case ' + str(n))
        remplir(n - 2)
```

Trace pour `n = 3` : la fonction affiche, dans l'ordre, `Remplir case 1`, `Remplir case 2`, `Vider case 1`, `Remplir case 3`, puis `Remplir case 1`. En partant de trois cases vides, ces coups légaux mènent à trois cases remplies.

RÉPONSE À RETENIR

```
def remplir(n):
    if n == 1:
        print('Remplir case 1')
    elif n > 1:
        remplir(n - 1)
        vider(n - 2)
        print('Remplir case ' + str(n))
        remplir(n - 2)
```

Question 10.

Sujet p. 11

> On envisage d'utiliser la fonction vider pour un baguenaudier de 2000 cases. Expliquer si ce code est adapté à un baguenaudier de si grande taille et justifier votre réponse.

Ce code n'est pas adapté à 2000 cases. La fonction appelle notamment `vider(n - 1)` : une chaîne d'appels peut donc atteindre une profondeur de l'ordre de `n`. Pour `n = 2000`, elle risque de dépasser la limite de récursion habituelle de Python et de provoquer une erreur de récursion ; la limite exacte dépend de l'environnement.

Le nombre d’actions affichées est lui aussi très grand. Si `a(n)` désigne le nombre d’affichages nécessaires pour vider ou remplir `n` cases, les deux procédures inverses ont le même nombre d’actions. On a `a(0) = 0`, `a(1) = 1` et, pour `n > 1`, `a(n) = a(n - 1) + 2a(n - 2) + 1`, d’après les appels récursifs de `vider`. Cette récurrence donne `a(n) = (4 × 2^n − (−1)^n − 3) / 6`, expression vérifiable avec les deux valeurs initiales et la récurrence. Ainsi, pour 2000 cases, `a(2000) = (2^2002 − 4) / 6` : le nombre d’actions est exponentiel.

On peut remplacer les appels récursifs par un traitement itératif utilisant une pile explicite d’instructions ; cela évite la limite de profondeur de la pile d’appels de Python. Mais cela ne réduit pas le nombre exponentiel d’actions : produire toute la résolution resterait impraticable pour 2000 cases.

> **RÉPONSE À RETENIR**
>
> **Non.** La profondeur de récursion risque de dépasser la limite de Python et le nombre d’actions à afficher croît exponentiellement. Un algorithme itératif avec une pile explicite peut éviter le dépassement de la pile d’appels, mais ne rend pas réalisable l’affichage de toutes les actions pour 2000 cases.

Exercice 3

8 points

Programmation Python, bases de données relationnelles, SQL, systèmes d'exploitation et sécurisation des communications

A

Question 1.

Sujet p. 13, 14

> Alice s'inscrit sur un nouveau site lui demandant de créer un mot de passe de 8 caractères minimum, composé uniquement de majuscules et de minuscules. Écrire un appel à la fonction gen_mdp qui permette de répondre aux exigences de ce site.

Le premier paramètre fixe la longueur à 8 caractères. Pour autoriser uniquement les minuscules et les majuscules, on indique `True` pour ces deux catégories et `False` pour les caractères spéciaux. Une longueur de 8 respecte le minimum demandé.

RÉPONSE À RETENIR

L'appel convenable est :

```
gen_mdp(8, True, True, False)
```

Question 2.

Sujet p. 12, 13, 14

> Recopier et compléter les lignes 8 à 10 du code de la fonction gen_mdp afin d'initialiser les variables minuscules, majuscules et caracteres_speciaux. Utiliser obligatoirement la syntaxe de tableaux donnés en compréhension. Après l'initialisation, minuscules doit contenir les lettres 'a' à 'z', majuscules les lettres 'A' à 'Z', et caracteres_speciaux les caractères '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?', '@'.

Dans la table ASCII, les codes des lettres minuscules vont de 97 à 122 et ceux des majuscules de 65 à 90. Les caractères spéciaux encadrés correspondent aux codes de 33 à 47, puis de 58 à 64. Comme la borne finale de `range` est exclue, on ajoute 1 à chacune des bornes finales.

RÉPONSE À RETENIR

```
minuscules = [chr(i) for i in range(97, 123)]
majuscules = [chr(i) for i in range(65, 91)]
caracteres_speciaux = [chr(i) for i in range(33, 48)] + [chr(i) for i in range(58,
65)]
```

Les deux intervalles de caractères spéciaux donnent respectivement `!` à `/` et `:` à `@`, soit exactement les caractères demandés.

Question 3.

Sujet p. 13, 14

> La variable jeu_caracteres est une liste contenant tous les caractères autorisés pour fabriquer un nouveau mot de passe. Recopier et compléter la ligne 13 du code de la fonction gen_mdp, puis écrire les autres lignes de code nécessaires à la création de jeu_caracteres. Le nombre de lignes de code à insérer est laissé à l'appréciation du

candidat.

La liste `jeu_caracteres` commence vide. Pour chaque catégorie autorisée par les paramètres, on lui ajoute la liste correspondante. La concaténation de listes avec + conserve tous les caractères autorisés.

RÉPONSE À RETENIR

```
jeu_caracteres = []
if cont_min:
    jeu_caracteres = jeu_caracteres + minuscules
if cont_maj:
    jeu_caracteres = jeu_caracteres + majuscules
if cont_spe:
    jeu_caracteres = jeu_caracteres + caracteres_speciaux
```

Question 4.

Sujet p. 13, 14

La documentation de la fonction randint indique que randint(a, b) renvoie un entier aléatoire compris entre a et b, extrémités incluses. Recopier et compléter la ligne 21 du code de la fonction gen_mdp.

Les indices d'une liste de longueur `n` vont de 0 à `n - 1`. On tire donc un indice dans cet intervalle, puis on ajoute le caractère choisi au mot de passe déjà construit.

RÉPONSE À RETENIR

```
mot_de_passe = mot_de_passe + jeu_caracteres[randint(0, n - 1)]
```

Question 5.

Sujet p. 13, 14

Un site demande un mot de passe contenant au moins 12 caractères, dont obligatoirement au moins un caractère spécial et au moins une lettre minuscule. Expliquer en quoi le code de la fonction gen_mdp peut renvoyer un mot de passe qui ne répond pas aux exigences du site.

Les paramètres `cont_min` et `cont_spe` indiquent que ces catégories sont autorisées, mais les caractères sont tirés au hasard dans une liste commune. La fonction ne garantit donc pas qu'au moins un caractère de chacune de ces catégories apparaisse. Par ailleurs, c'est l'argument `longueur` qui fixe le nombre de caractères : la fonction ne vérifie pas à elle seule qu'il est au moins égal à 12.

RÉPONSE À RETENIR

Par exemple, avec une longueur de 12 et les minuscules et caractères spéciaux autorisés, le tirage peut produire un mot de passe composé uniquement de minuscules, ou uniquement de caractères spéciaux. Il ne respecterait alors pas les exigences de composition. Il faut aussi appeler la fonction avec une longueur d'au moins 12 et ajouter une vérification ou une construction qui impose la présence d'au moins un caractère de chaque catégorie requise.

B

Question 6.

Sujet p. 15, 16

> Expliquer en quoi il n'est pas possible, pour Alice, d'avoir le même mot de passe pour deux sites différents.

`mot_de_passe` est la clé primaire de la table `compte`. Une clé primaire doit être unique dans la table : deux lignes ne peuvent pas avoir la même valeur pour cet attribut. Cette contrainte s'applique donc aussi à des comptes de sites différents.

RÉPONSE À RETENIR

La contrainte de clé primaire sur `compte.mot_de_passe` interdit à Alice d'enregistrer le même mot de passe pour deux sites différents.

Question 7.

Sujet p. 15, 16

> Écrire la requête SQL permettant d'afficher toutes les URL enregistrées dans la base de données.

Les URL sont stockées dans l'attribut `url` de la table `site`. Il suffit donc de sélectionner cet attribut dans cette table.

RÉPONSE À RETENIR

```
SELECT url
FROM site;
```

Sur les extraits fournis, la requête renvoie :

url
https://logi-educ.net/vosnotes/eleve.html
https://www.banqueperso.fr/connexion.html
https://espace-client.ev.fr/login
https://account.votremailp.me/login

Question 8.

Sujet p. 15, 16

> La Banque Perso a demandé à Alice de renouveler son mot de passe. Elle remplace le mot de passe '@rDfohpj!&' par 'yhTS?d@UTJe'. Écrire la requête SQL permettant d'effectuer la modification dans la base de données. Ignorer l'attribut renouvellement et ne pas chercher à le mettre à jour.

L'ancien mot de passe identifie la ligne à modifier puisqu'il est unique, étant la clé primaire. On met à jour uniquement cet attribut, comme demandé.

RÉPONSE À RETENIR

```
UPDATE compte
SET mot_de_passe = 'yhTS?d@UTJe'
WHERE mot_de_passe = '@rDfohpj!&';
```

Question 9.

Sujet p. 15, 16

> À la date du 20 mars 2025, Alice a décidé de lister tous les mots de passe qui n'ont pas été renouvelés depuis plus d'un an. Écrire la requête SQL permettant de donner la liste des id_site concernés. Le langage SQL utilise l'ordre lexicographique pour comparer deux éléments de type texte.

Au 20 mars 2025, une date strictement antérieure au 20 mars 2024 correspond à un renouvellement datant de plus d'un an. Avec le format `AAAA-MM-JJ`, la comparaison lexicographique des chaînes donne l'ordre chronologique. `DISTINCT` évite de répéter un site si plusieurs mots de passe le concernent.

RÉPONSE À RETENIR

```
SELECT DISTINCT id_site
FROM compte
WHERE renouvellement < '2024-03-20'
ORDER BY id_site;
```

Sur les données fournies, toutes les dates sont antérieures à cette limite. Les identifiants de site distincts sont :

id_site
1
2
3
4

Question 10.

Sujet p. 15, 16

> Donner une raison pour laquelle Alice a préféré choisir le format AAAA-MM-JJ pour l'attribut renouvellement plutôt que le format JJ-MM-AAAA.

Avec le format `AAAA-MM-JJ`, l'année, puis le mois et le jour sont écrits dans cet ordre. C'est aussi l'ordre de priorité nécessaire pour comparer chronologiquement deux dates sous forme de chaînes.

RÉPONSE À RETENIR

Ce format permet de trier des dates stockées comme chaînes dans l'ordre chronologique par ordre lexicographique, contrairement au format `JJ-MM-AAAA`.

Question 11.

Sujet p. 15, 16

> Écrire une requête SQL permettant d'afficher tous les utilisateurs et les mots de passe du ou des sites de nom 'Votremailp', dont l'identifiant est supposé non connu. Le résultat devra être affiché par ordre chronologique de date de renouvellement du mot de passe.

On relie les tables par la clé étrangère `compte.id_site` et la clé primaire `site.id`, puis on filtre sur le nom du site et on trie les comptes par date de renouvellement croissante.

RÉPONSE À RETENIR

```
SELECT compte.utilisateur, compte.mot_de_passe
FROM compte
JOIN site ON compte.id_site = site.id
WHERE site.nom_site = 'Votremailp'
ORDER BY compte.renouvellement ASC;
```

Sur les extraits fournis, les deux lignes sont ordonnées par date de renouvellement comme suit : la première date est `2018-10-14`, puis la date `2022-06-30`.

utilisateur	mot_de_passe
alice_B@votremailp.me	GxRGDxc(u-PM
alice_B2@votremailp.me	cYFgt!:Ehr;

Question 12.

Sujet p. 15, 16

> Pour réaliser le projet, il aurait été possible de regrouper toutes les informations dans une seule table, mais Alice a choisi d'utiliser deux tables : site et compte. Donner un avantage impliqué par le choix d'Alice.

Dans une base séparée en deux tables, les informations propres à un site ne sont pas répétées dans chaque ligne de compte associée à ce site.

RÉPONSE À RETENIR

Un avantage est la réduction de la redondance : le nom et l'URL d'un site sont stockés une seule fois dans la table `site`. Cela limite aussi les risques d'incohérence lorsqu'une information sur ce site est modifiée.

C

Question 13.

Sujet p. 16, 17

> Alice ouvre un interpréteur Python dans le répertoire Documents, qui est le répertoire courant, et exécute le contenu du fichier chiffrement.py. Écrire l'appel à la fonction chiffrement permettant, à l'aide de la clé contenue dans le fichier cle, de chiffrer le fichier gestionnaire.db en créant le fichier chiffré secret.db dans le répertoire Perso.

Le répertoire courant est `Documents`. Le fichier source s'y trouve directement. D'après l'arborescence, `Perso` est son répertoire frère : on y accède en remontant d'un niveau avec `..`, puis en entrant dans `Perso`. La clé et le fichier chiffré doivent donc être désignés par des chemins relatifs commençant par `../Perso/`.

RÉPONSE À RETENIR

```
chiffrement("gestionnaire.db", "../Perso/secret.db", "../Perso/cle")
```

Question 14.

Sujet p. 18

> Donner le résultat, en écriture hexadécimale, de l'opération OU-exclusif bit à bit entre deux octets dont les écritures hexadécimales sont respectivement A3 et 59.

On écrit les deux octets sur 8 bits, puis on applique XOR bit à bit : `A3` vaut `1010 0011` et `59` vaut `0101 1001`. Le résultat est `1111 1010`, qui correspond à `FA` en hexadécimal.

RÉPONSE À RETENIR

`A3 XOR 59 = FA`.

Question 15.

Sujet p. 18

> Montrer que, pour tous bits a et b, on a la propriété suivante : (a XOR b) XOR b = a.

Les bits ne peuvent prendre que les valeurs 0 et 1. La table suivante examine les quatre couples possibles et montre que le résultat final est toujours égal à `a`.

a	b	a XOR b	(a XOR b) XOR b
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

RÉPONSE À RETENIR

Dans les quatre cas, la dernière colonne est égale à la première. Ainsi, pour tous bits `a` et `b`, `(a XOR b) XOR b = a`.

Question 16.

Sujet p. 17, 18

> Indiquer, en justifiant, si le chiffrement utilisé par Alice est symétrique ou asymétrique.

Le fichier chiffré est obtenu en appliquant XOR avec la clé. Pour retrouver le fichier source, il suffit d'appliquer à nouveau XOR avec cette même clé : la propriété démontrée à la question précédente donne `(a XOR b) XOR b = a`.

RÉPONSE À RETENIR

Ce chiffrement est symétrique : la même clé sert au chiffrement et au déchiffrement. Un chiffrement asymétrique utilise une paire de clés différentes.

Question 17.

Sujet p. 16, 18

> Alice exécute la commande ls -l secret.db et obtient : -rw-r--r-- 1 alice eleves 42480 mars 25 11:57 secret.db. Justifier le fait qu'un attaquant a le champ libre pour tenter un déchiffrement du fichier secret.db et expliquer ce qu'Alice devrait faire pour corriger ce problème.

Dans `-rw-r--r--`, les droits sont `rw-` pour la propriétaire, `r--` pour le groupe et `r--` pour les autres. Le fichier est donc lisible par les autres utilisateurs qui peuvent aussi traverser les répertoires de son chemin. Ils peuvent récupérer une copie du fichier chiffré et tenter des essais de déchiffrement hors ligne.

> **RÉPONSE À RETENIR**
>
> Alice devrait retirer les droits de lecture au groupe et aux autres utilisateurs, par exemple avec :
>
> ```
> chmod 600 secret.db
> ```
>
> Elle doit également protéger le fichier de clé avec des droits stricts et veiller à ce que les répertoires du chemin ne soient pas accessibles aux autres. Les permissions limitent l'accès aux fichiers, mais ne remplacent pas un algorithme de chiffrement robuste.

D

Question 18.

Sujet p. 13, 14, 15, 16, 17, 18, 19

> En justifiant votre réponse, préciser en quoi les choix d'Alice dans la conception de son gestionnaire de mots de passe respectent ou non chacune des quatre bonnes pratiques mentionnées : P1, utiliser un mot de passe différent pour chaque accès ; P2, créer un mot de passe suffisamment long, complexe et inattendu, d'au moins 8 caractères, contenant des minuscules, des majuscules, des chiffres et des caractères spéciaux ; P3, ne jamais communiquer son mot de passe à un tiers ; P4, utiliser un gestionnaire de mots de passe qui mémorise les mots de passe et permet d'en générer aléatoirement.

P1 — Un mot de passe différent pour chaque accès : la clé primaire `compte.mot_de_passe` interdit les doublons dans la table. Les mots de passe enregistrés pour les différents accès sont donc distincts.

P2 — Un mot de passe long, complexe, inattendu, avec minuscules, majuscules, chiffres et caractères spéciaux : la fonction permet de choisir la longueur et d'autoriser minuscules, majuscules et caractères spéciaux, mais elle ne propose pas de chiffres. De plus, autoriser une catégorie ne garantit pas qu'un caractère de cette catégorie sera effectivement tiré. Elle ne garantit donc pas à elle seule toutes les caractéristiques de P2.

P3 — Ne jamais communiquer son mot de passe à un tiers : l'énoncé indique qu'Alice crée un gestionnaire pour stocker ses informations de connexion, mais ne dit pas qu'elle communique ses mots de passe à un tiers. On ne peut donc pas conclure à un manquement à cette pratique à partir des informations fournies.

P4 — Utiliser un gestionnaire qui mémorise les mots de passe et permet d'en générer aléatoirement : le projet d'Alice stocke les informations de connexion dans une base et prévoit une fonction qui produit des mots de passe par tirages aléatoires. Le principe répond à P4, même si le générateur doit être amélioré pour garantir les critères de composition demandés.

RÉPONSE À RETENIR

Les choix d’Alice respectent P1 par la contrainte d’unicité de la clé primaire et vont dans le sens de P4 grâce au stockage et à la génération aléatoire. La fonction actuelle ne satisfait pas entièrement P2 : elle ne prévoit pas de chiffres et ne garantit pas la présence de chaque catégorie autorisée. Pour P3, l’énoncé ne permet pas de savoir si Alice communique ses mots de passe à un tiers.

Sujet : https://www.sujetdebac.fr/annales-pdf/2025/spe-numerique-informatique-2025-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.