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

NSI

Session 2026 • Antilles-Guyane • Jour 2

Code sujet **Non indiqué dans le document** Non indiqué dans le document

Proposition de corrigé pédagogique • FormaV

Se repérer dans le corrigé

Exercice 1 — Bases de données, langage SQL et POO
11 réponses • sujet p. 1–3

Exercice 2 — Réseaux, routage et graphes
11 réponses • sujet p. 4–6

Exercice 3 — Tableaux, dictionnaires, récursivité et programmation dynamique
16 réponses • sujet p. 7–9

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

Bases de données, langage SQL et POO

A

Question 1.

Sujet p. 1

> Donner, en justifiant, les clés étrangères de la relation reservation.

Une clé étrangère est un attribut qui référence la clé primaire d'une autre relation. La relation `reservation` contient les identifiants du client et de la salle associés à chaque réservation : `id_client` correspond à la clé primaire `client.id_client`, et `id_salle` à la clé primaire `salle.id_salle`.

L'attribut `id` de `reservation` est sa clé primaire, pas une clé étrangère. Les extraits de tables peuvent être incomplets : l'absence d'un client dans l'extrait ne remet donc pas en cause la référence.

RÉPONSE À RETENIR

Les clés étrangères de `reservation` sont `id_client`, qui référence `client(id_client)`, et `id_salle`, qui référence `salle(id_salle)`.

Question 2.

Sujet p. 1

> Donner le résultat de la requête SELECT intitule FROM salle WHERE nb_places > 5 ORDER BY tarif_jour lorsqu'elle est appliquée aux extraits des tables fournis.

La condition `nb_places > 5` retient les salles Ravel (90 places, tarif 2300), RA (10 places, tarif 350) et RB (8 places, tarif 300). La clause `ORDER BY tarif_jour` trie par tarif croissant par défaut : RB, puis RA, puis Ravel.

RÉPONSE À RETENIR

intitule
RB
RA
Ravel

Question 3.

Sujet p. 1, 2

> Écrire une requête permettant d'obtenir la liste sans doublon des id_salle des salles réservées depuis janvier 2024. On pourra utiliser le comparateur > pour les dates.

On sélectionne les identifiants de salle dans `reservation`, car cette relation indique les salles réservées et leurs dates. La date de début de janvier 2024 est le 1er janvier 2024 ; avec des dates au format ISO, on peut utiliser le comparateur demandé et écrire `date > '2023-12-31'`. `DISTINCT` élimine les doublons. Sur l'extrait, les réservations correspondantes concernent les salles 3, 2 et 1. Sans `ORDER BY`, SQL ne garantit pas l'ordre d'affichage.

RÉPONSE À RETENIR

```
SELECT DISTINCT id_salle
FROM reservation
WHERE date > '2023-12-31';
```

Résultat sur l'extrait : les identifiants obtenus sont 3, 2 et 1, soit l'ensemble {1, 2, 3}. Leur ordre d'affichage n'est pas garanti.

Question 4.

Sujet p. 1, 2

> Un nouveau client de nom de société EN et de numéro de téléphone 0144624055 a réservé le bureau d'identifiant 4 pour une journée le 12 juin 2024. Écrire une suite de requêtes permettant d'ajouter ce client et sa réservation. On attribuera la valeur 3 à id_client et la valeur 5 à id.

Il faut insérer le client avant sa réservation afin que l'identifiant du client soit présent pour la référence de réservation. Les identifiants `id_client = 3` et `id = 5` sont ceux imposés par l'énoncé. On enregistre le numéro de téléphone comme une chaîne afin de conserver son zéro initial. La durée indiquée est d'une journée.

RÉPONSE À RETENIR

```
INSERT INTO client (id_client, societe, num_tel)
VALUES (3, 'EN', '0144624055');

INSERT INTO reservation (id, id_client, id_salle, date, duree)
VALUES (5, 3, 4, '2024-06-12', 1);
```

Question 5.

Sujet p. 1, 2

> La salle RAVEL a été modifiée : elle peut maintenant contenir 95 personnes. Écrire la requête permettant d'enregistrer cette modification dans la base de données précédente.

La modification porte sur le nombre de places de la salle dont l'intitulé est `Ravel` dans l'extrait fourni. La clause `WHERE` limite la mise à jour à cette salle.

RÉPONSE À RETENIR

```
UPDATE salle
SET nb_places = 95
WHERE intitule = 'Ravel';
```

Question 6.

Sujet p. 1, 2

> Écrire une requête permettant d'obtenir le temps total de réservation de la salle d'identifiant 6.

On additionne les durées de toutes les réservations dont `id_salle` vaut 6. La fonction d'agrégation `SUM` convient. Dans l'extrait fourni, aucune réservation ne concerne la salle 6 ; en SQL, une somme portant sur un ensemble vide renvoie `NULL` (et non 0).

RÉPONSE À RETENIR

```
SELECT SUM(duree) AS temps_total
FROM reservation
WHERE id_salle = 6;
```

Résultat sur l'extrait : `NULL`, car aucune ligne de réservation pour la salle 6 n'y figure.

Question 7.

Sujet p. 1, 2

> Écrire une requête permettant d'obtenir la liste sans doublon des noms de société ayant réservé une salle de réunion.

Pour obtenir les sociétés, il faut relier les réservations aux clients par leur identifiant de client et aux salles par leur identifiant de salle. On filtre ensuite les salles dont le type est `Réunion`. `DISTINCT` garantit une liste sans doublon.

Dans l'extrait, la réservation de la salle 3, de type Réunion, est celle du client 2, dont la société est FleursJ. Aucune réservation de la salle 5 n'apparaît dans l'extrait.

RÉPONSE À RETENIR

```
SELECT DISTINCT client.societe
FROM reservation
JOIN client ON reservation.id_client = client.id_client
JOIN salle ON reservation.id_salle = salle.id_salle
WHERE salle.type = 'Réunion';
```

Résultat sur l'extrait : FleursJ.

B

Question 8.

Sujet p. 2

> Donner un exemple d'attribut et un exemple de méthode de la classe Salle.

Un attribut est une donnée mémorisée dans chaque objet de la classe ; une méthode est une fonction définie dans la classe et appelée sur un objet. Dans le code, `intitule` est un attribut et `reserver` est une méthode.

RÉPONSE À RETENIR

Exemple d'attribut : `self.intitule`. Exemple de méthode : `reserver(self, client)`.

Question 9.

Sujet p. 2

> Écrire la méthode liberer qui modifie les attributs d'une salle de manière à la rendre disponible et sans occupant (chaîne de caractères vide).

Pour libérer une salle, la méthode doit remettre son indicateur de disponibilité à `True` et vider l'attribut `occupant`. Comme elle agit sur l'objet, elle reçoit `self`.

RÉPONSE À RETENIR

```
def liberer(self):
    self.dispo = True
    self.occupant = ""
```

Question 10.

Sujet p. 2

> Recopier et compléter les lignes 3, 4 et 5 du code de la fonction salles_dispos, qui prend en paramètre une liste de salles et renvoie la liste des intitulés de celles qui sont disponibles.

On parcourt les objets salle de la liste. Pour chaque salle disponible, on ajoute son intitulé à la liste résultat. Dans l'exemple donné, `s1` et `s2` ont été réservées ; les intitulés des salles disponibles sont donc RA, B2 et RB.

RÉPONSE À RETENIR

```
def salles_dispos(liste_salles):
    res = []
    for salle in liste_salles:
        if salle.dispo:
            res.append(salle.intitule)
    return res
```

Avec la liste de l'énoncé, la fonction renvoie `['RA', 'B2', 'RB']`.

Question 11.

Sujet p. 3

> Recopier et compléter la méthode ajouter_reservation, qui ajoute salle à la liste des salles réservées, puis réserve cette salle en modifiant le prix total. Elle renvoie True si la salle a pu être réservée car elle était libre, et False sinon.

Si `salle.dispo` vaut `True`, la salle peut être ajoutée à la liste du séminaire et réservée au nom du client. La méthode `salle.reserver` rend le tarif de la salle et met à jour sa disponibilité ainsi que son occupant ; ce tarif est ajouté au prix total. Si la salle n'est pas disponible, on ne modifie rien et on renvoie `False`.

RÉPONSE À RETENIR

```
def ajouter_reservation(self, salle):
    if salle.dispo:
        self.liste_salles.append(salle)
        self.prix_total += salle.reserver(self.client)
        return True
    return False
```

Le test préalable garantit que `salle.reserver(self.client)` renvoie le tarif, et non `False`, lors de cette réservation.

Exercice 2

6 points

Réseaux, routage et graphes

A

Question 1.

Sujet p. 4

> Expliquer pourquoi le routeur R2 ne peut pas être associé à l'adresse IP 192.168.2.3 dans le sous-réseau qui le lie au routeur R1.

Le lien entre R1 et R2 appartient au sous-réseau `192.168.1.0/24`. Avec un masque `/24`, une adresse de ce sous-réseau doit avoir les trois premiers octets `192.168.1`. Or `192.168.2.3` appartient au sous-réseau `192.168.2.0/24`, qui relie R4 à R6.

RÉPONSE À RETENIR

R2 ne peut pas utiliser `192.168.2.3` sur le lien R1–R2, car cette adresse n'appartient pas au sous-réseau `192.168.1.0/24` de ce lien.

Question 2.

Sujet p. 4

> Recopier et compléter la table de routage du routeur R4 avec les adresses IP des destinations et des passerelles (ou routeurs suivants) selon le protocole RIP, après l'échange avec l'ensemble des routeurs. Une destination correspond à un sous-réseau. La table fournie indique déjà : 10.1.6.0/24, connecté, distance 0 ; 10.2.6.0/24, connecté, distance 0 ; 192.168.2.0/24, connecté, distance 0 ; 10.1.1.0/24, passerelle 10.1.6.1, distance 1.

R4 est directement connecté aux sous-réseaux indiqués comme connectés dans la table. Selon RIP, les distances comptent les routeurs à traverser. Le sous-réseau `10.1.1.0/24`, relié à R3, est atteint depuis R4 en passant par R3 : sa distance est 1 et la passerelle est l'adresse de R3 sur le lien R3–R4, soit `10.1.6.1`. Pour atteindre les sous-réseaux reliés à R2, il faut ensuite passer par R2 : la distance est donc 2. Le sous-réseau `10.2.7.0/24` est accessible en un saut par R5 ou par R6 ; la table ci-dessous choisit R5 comme passerelle.

RÉPONSE À RETENIR

Destination	Passerelle	Distance
10.1.6.0/24	connecté	0
10.2.6.0/24	connecté	0
192.168.2.0/24	connecté	0
10.1.1.0/24	10.1.6.1	1
192.168.1.0/24	10.1.6.1	2
192.168.4.0/24	10.1.6.1	2
10.2.7.0/24	10.2.6.2	1

Pour `10.2.7.0/24`, la passerelle `192.168.2.1` (R6) est également possible avec la même distance 1.

Question 3.

Sujet p. 4

> Les routeurs R3 et R4 sont reliés par la technologie 5G, dont le débit est de 10 Gbit/s. Calculer le débit entre les routeurs R1 et R2. On rappelle que le coût d'un lien est inversement proportionnel au débit de la liaison.

Si le coût est inversement proportionnel au débit, le produit coût × débit est constant pour les liens concernés. Le coût du lien R3–R4 vaut 1 et son débit est de 10 Gbit/s. Le coût du lien R1–R2 vaut 20. On obtient donc :

$1 \times 10 = 20 \times d$, donc $d = \frac{10}{20} = 0{,}5\,\mathrm{Gbit/s}$.

RÉPONSE À RETENIR

Le débit entre R1 et R2 est de **0,5 Gbit/s**.

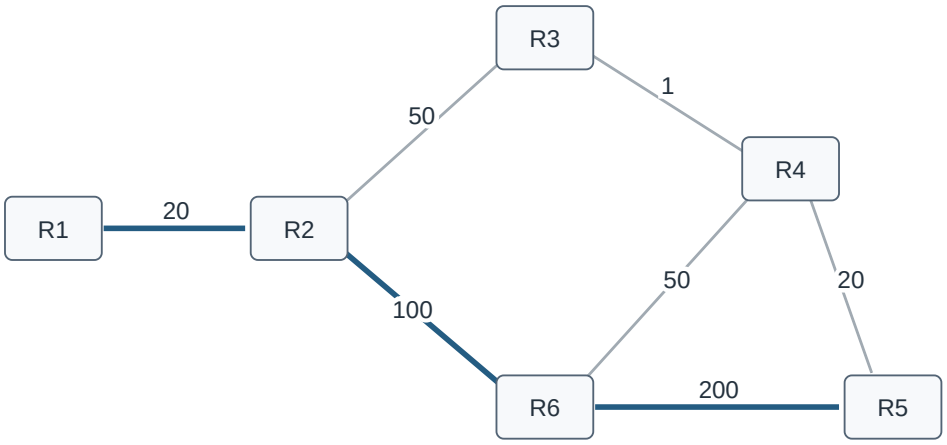
Question 4.

Sujet p. 4

> Indiquer le chemin à suivre pour transmettre des données de R1 à R5 selon le protocole RIP.

RIP minimise le nombre de routeurs à traverser, donc le nombre de liens du chemin. Le trajet R1–R2–R6–R5 utilise 3 liens. Le trajet par R3 et R4, R1–R2–R3–R4–R5, en utilise 4. Le premier est donc le chemin retenu par RIP.

Réseau pondéré — chemin RIP de R1 à R5



Les nombres indiquent les coûts des liens de la figure 1. Le chemin RIP retenu est mis en évidence.

RÉPONSE À RETENIR

Le chemin est **R1 → R2 → R6 → R5**.

Question 5.

Sujet p. 4, 5

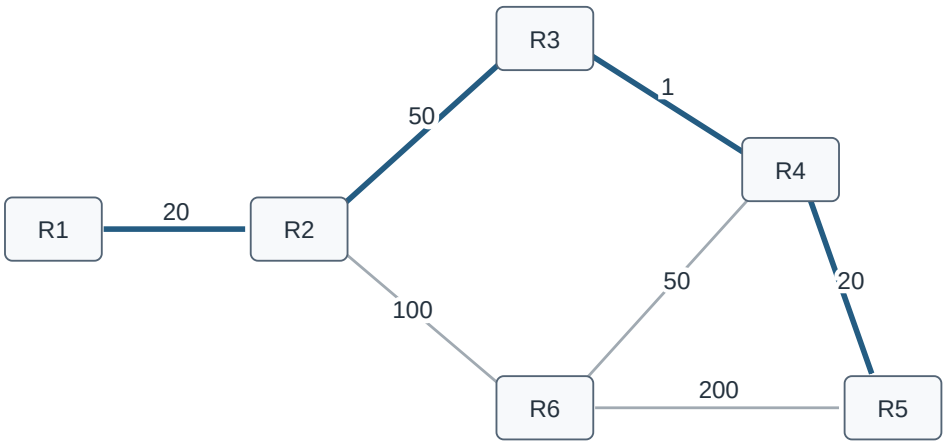
> La liaison R2-R6 tombe en panne. Indiquer le chemin à suivre pour transmettre des données de R1 à R5, toujours

FormaV - document d'entraînement non officiel

selon le protocole RIP.

La liaison R2–R6 est en panne : le chemin R1–R2–R6–R5 n'est donc plus disponible. Depuis R2, le trajet restant vers R5 passe par R3 puis R4. Il compte 4 liens depuis R1.

Réseau pondéré — chemin RIP avec la liaison R2–R6 en panne



Les nombres indiquent les coûts de la figure 1. La liaison R2–R6, dessinée pour conserver le graphe d'origine, est en panne et ne peut pas être empruntée. Le chemin retenu est mis en évidence.

RÉPONSE À RETENIR

Le chemin est **R1 → R2 → R3 → R4 → R5**.

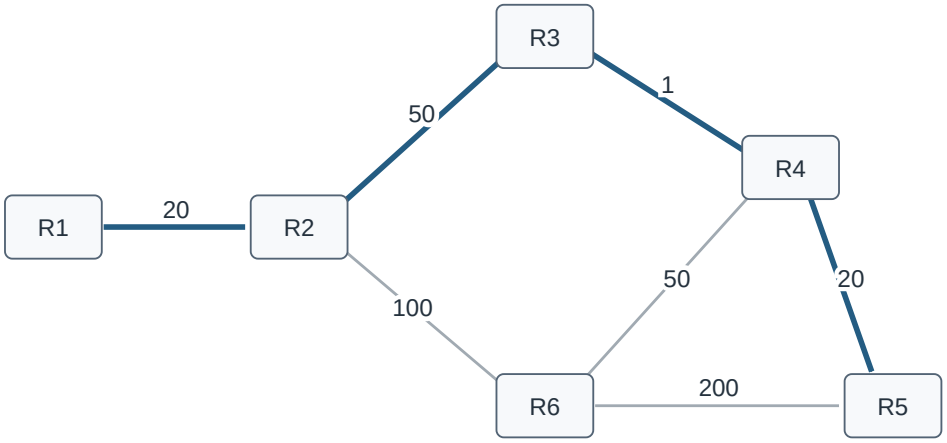
Question 6.

Sujet p. 4, 5

La liaison R2-R6 est rétablie et le protocole OSPF est utilisé. Dans ce protocole, la métrique de la table de routage correspond à la somme des coûts des liaisons traversées. Indiquer le chemin à suivre pour transmettre des données de R1 à R5 selon OSPF et le comparer avec le chemin obtenu précédemment selon RIP.

Avec OSPF, on minimise la somme des coûts. Pour R1–R2–R3–R4–R5, le coût vaut `20 + 50 + 1 + 20 = 91`. Les principaux autres chemins ont des coûts plus élevés : R1–R2–R6–R5 coûte `20 + 100 + 200 = 320`, et R1–R2–R6–R4–R5 coûte `20 + 100 + 50 + 20 = 190`. Le chemin de coût 91 est donc le chemin OSPF. Sur le réseau rétabli, RIP choisissait R1–R2–R6–R5, qui comporte 3 liens : les deux protocoles donnent ici des chemins différents.

Réseau pondéré — chemin OSPF de R1 à R5



Les nombres indiquent les coûts exacts de la figure 1. Le chemin OSPF de coût total 91 est mis en évidence.

> **RÉPONSE À RETENIR**
>
> Selon OSPF, le chemin est **R1 → R2 → R3 → R4 → R5**, de coût total **91**. Il diffère du chemin RIP **R1 → R2 → R6 → R5**.

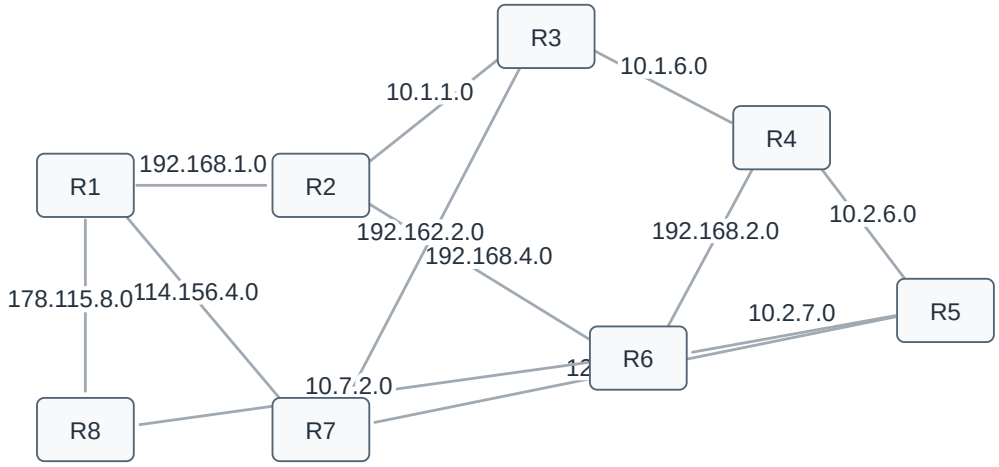
Question 7.

Sujet p. 5

> Le réseau évolue et de nouveaux routeurs sont implantés. Pour des raisons de lisibilité, le suffixe /24 est omis dans le tableau récapitulant les mises à jour des adresses IP des routeurs. Représenter le réseau informatique mis à jour à l'aide d'un graphe.

Chaque sous-réseau /24 partagé par deux routeurs forme une arête du graphe. Par exemple, R1 et R7 partagent le sous-réseau `114.156.4.0/24` ; ils sont donc reliés. Le tableau ne donne pas de coûts pour ces liens : le graphe représente uniquement les connexions.

Graphe du réseau informatique mis à jour



Chaque arête correspond au sous-réseau /24 indiqué par les adresses IP du tableau.

RÉPONSE À RETENIR

Le réseau mis à jour est représenté ci-dessous ; les étiquettes indiquent les sous-réseaux partagés.

B

Question 8.

Sujet p. 5

Un automobiliste part de V1 pour rejoindre V5. Le GPS propose l'itinéraire A, qui minimise le temps de parcours, et l'itinéraire B, qui minimise le nombre de villes traversées. Pour ces deux itinéraires, indiquer celui qui correspond au protocole RIP et celui qui correspond au protocole OSPF d'un réseau informatique.

RIP choisit le chemin ayant le moins de sauts, ce qui correspond à minimiser le nombre de villes traversées : c'est l'itinéraire B. OSPF minimise la somme des coûts des liaisons ; dans l'analogie avec le GPS, cela correspond à minimiser le temps de parcours : c'est l'itinéraire A.

RÉPONSE À RETENIR

L'itinéraire B correspond à RIP ; **l'itinéraire A** correspond à OSPF.

Question 9.

Sujet p. 5, 6

Écrire la liste des sommets, dans l'ordre de leur visite, lors du parcours en largeur du graphe du réseau routier de la figure 2 depuis le sommet 'V3'.

On effectue le parcours en largeur en respectant l'ordre des voisins indiqué dans le dictionnaire. À partir de V3, on visite d'abord V2 puis V4. En poursuivant niveau par niveau, on visite ensuite V1, V6 puis V5.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

`['V3', 'V2', 'V4', 'V1', 'V6', 'V5']`

Question 10.

Sujet p. 5, 6

> Compléter l'évolution du contenu de la variable precedent à chaque tour de la boucle while lors de l'exécution de itineraire_court(route, 'V1', 'V5'). Les contenus fournis sont : tour 1, {'V1': None} ; tour 2, {'V1': None, 'V2': 'V1'} ; tour 3, {'V1': None, 'V2': 'V1', 'V3': 'V2', 'V6': 'V2'} ; puis […].

Les trois états donnés numérotent d'abord le dictionnaire initial comme tour 1, puis l'état après chaque passage dans la boucle `while`. Après le tour 3 fourni, la file contient `['V3', 'V6']`. On traite V3 : V4 est ajouté avec V3 comme précédent. Au tour suivant, on traite V6. V4 n'ayant pas encore été visité, le code remplace son précédent par V6 ; il ajoute aussi V5 avec V6 comme précédent. Comme V5 est l'arrivée, la file est vidée et la boucle s'arrête.

RÉPONSE À RETENIR

Tour n°4 : `{'V1': None, 'V2': 'V1', 'V3': 'V2', 'V6': 'V2', 'V4': 'V3'}`

Tour n°5 : `{'V1': None, 'V2': 'V1', 'V3': 'V2', 'V6': 'V2', 'V4': 'V6', 'V5': 'V6'}`

Il n'y a pas de tour suivant : la découverte de V5 vide la file.

Question 11.

Sujet p. 5, 6

> Écrire ce que renvoie la fonction itineraire_court à la suite de l'exécution de itineraire_court(route, 'V1', 'V5').

À la fin, les précédents utiles pour reconstruire le chemin sont V5 ← V6, V6 ← V2 et V2 ← V1. En remontant depuis V5, on obtient V5, V6, V2, V1 ; la fonction inverse ensuite cette liste avec `reverse()`.

RÉPONSE À RETENIR

La fonction renvoie `['V1', 'V2', 'V6', 'V5']`.

Exercice 3

8 points

Tableaux, dictionnaires, récursivité et programmation dynamique

A

Question 1.

Sujet p. 7

> Donner une instruction Python qui permet de déclarer et d'initialiser le dictionnaire effectif correspondant à un stock constitué de trois planches de 1 m, de quatre planches de 2 m et d'une planche de 4 m.

Le dictionnaire associe à chaque longueur présente dans le stock son nombre de planches. On indique donc les longueurs 1, 2 et 4, avec les effectifs donnés.

> **RÉPONSE À RETENIR**
>
> `effectif = {1: 3, 2: 4, 4: 1}`

Question 2.

Sujet p. 7

> Donner une expression en Python qui calcule la valeur totale, en euro, de l'ensemble des planches de 10 m d'un stock représenté par le dictionnaire effectif contenant au moins une planche de 10 m. On rappelle que la variable prix contient l'ensemble des prix de vente en euro des planches.

On multiplie le nombre de planches de 10 m par le prix d'une planche de 10 m. La clé 10 est présente dans le dictionnaire, comme l'indique l'énoncé. Ici, `prix[10]` vaut 30.

> **RÉPONSE À RETENIR**
>
> `effectif[10] * prix[10]`, soit `effectif[10] * 30`.

Question 3.

Sujet p. 7

> Donner une expression en Python qui vaut True si le stock représenté par le dictionnaire effectif contient des planches de 5 m et False sinon.

Les clés du dictionnaire sont exactement les longueurs présentes dans le stock. Il suffit donc de tester si la clé 5 est présente.

> **RÉPONSE À RETENIR**
>
> `5 in effectif`

Question 4.

Sujet p. 7

> Recopier et compléter les lignes 2, 3 et 4 du code de la fonction valeur_stock, qui prend en paramètre le dictionnaire effectif et renvoie la valeur, en euro, du stock représenté par effectif si l'on vend toutes les planches selon les prix de la variable globale prix.

On parcourt les longueurs présentes dans le dictionnaire et on additionne, pour chacune, le produit de son effectif par le prix unitaire correspondant. L'initialisation à 0 convient aussi si le dictionnaire est vide.

FormaV - document d'entraînement non officiel

RÉPONSE À RETENIR

```
def valeur_stock(effectif):
    valeur_totale = 0
    for longueur in effectif:
        valeur_totale = valeur_totale + effectif[longueur] * prix[longueur]
    return valeur_totale
```

Question 5.

Sujet p. 7

> Écrire une fonction mise_a_jour qui prend en paramètre l'effectif du stock déjà présent dans la scierie, nommé effectif, et l'effectif du stock contenu dans le camion, nommé autre_effectif, et qui met à jour le dictionnaire effectif.

Pour chaque longueur présente dans le camion, on ajoute son effectif à celui du stock. Si cette longueur n'est pas encore dans le stock, `effectif.get(longueur, 0)` fournit un effectif précédent égal à 0. La fonction modifie le dictionnaire reçu. Dans l'exemple, l'effectif de longueur 1 devient 5, celui de longueur 2 devient 2, et celui de longueur 5 reste 5.

RÉPONSE À RETENIR

```
def mise_a_jour(effectif, autre_effectif):
    for longueur in autre_effectif:
        effectif[longueur] = effectif.get(longueur, 0) + autre_effectif[longueur]
```

Question 6.

Sujet p. 7

> Écrire une fonction vendre qui prend en paramètre l'effectif du stock, nommé effectif, et une longueur de planche, nommée longueur, et qui modifie le dictionnaire effectif en conséquence. On a déjà vérifié qu'il y avait une planche de cette longueur dans le stock. Si l'effectif de cette longueur devient nul, l'association devra être supprimée du dictionnaire effectif. On pourra utiliser del d[c] pour supprimer la clé c du dictionnaire d.

L'existence d'une planche de la longueur demandée a déjà été vérifiée. On diminue donc l'effectif de 1. Si l'effectif devient nul, on supprime l'association correspondante, car le dictionnaire ne contient que les longueurs présentes dans le stock.

RÉPONSE À RETENIR

```
def vendre(effectif, longueur):
    effectif[longueur] = effectif[longueur] - 1
    if effectif[longueur] == 0:
        del effectif[longueur]
```

B

Question 7.

Sujet p. 7, 8

> Déterminer, en justifiant, la valeur maximale que peut rapporter une planche de 4 m.

Les prix donnés sont `prix[1] = 1`, `prix[2] = 5`, `prix[3] = 8` et `prix[4] = 9`. Les découpages possibles d'une longueur totale de 4 donnent les sommes suivantes : vendre la planche entière rapporte 9 ; les découpages en deux morceaux, 1 + 3, 2 + 2 et 3 + 1, rapportent respectivement 9, 10 et 9 ; avec trois morceaux, 1 + 1 + 2 rapporte 7 ; avec quatre morceaux, 1 + 1 + 1 + 1 rapporte 4. La meilleure valeur est donc 10 euros.

> **RÉPONSE À RETENIR**
>
> La valeur maximale est **10 euros**, obtenue en découpant la planche en deux morceaux de 2 m : `prix[2] + prix[2] = 5 + 5 = 10`.

Question 8.

Sujet p. 8

> Expliquer le cas d'arrêt de la formule récursive proposée : si n ≤ 3, alors valmax(n) = prix[n].

Pour une longueur de 1 m, aucune découpe en morceaux de longueurs entières positives plus petites n'est possible. Pour 2 m et 3 m, l'énoncé montre que les découpages rapportent moins que la vente de la planche entière. On peut donc renvoyer directement `prix[n]` pour ces longueurs, sans poursuivre les appels récursifs.

> **RÉPONSE À RETENIR**
>
> Le cas d'arrêt `n ≤ 3` est justifié parce que, pour ces longueurs, vendre la planche entière rapporte au moins autant que toute découpe possible : `valmax(n) = prix[n]`.

Question 9.

Sujet p. 7, 8

> Recopier et compléter les quatre lignes suivantes : valmax(1) = prix[…] = … ; valmax(2) = … = … ; valmax(3) = … = … ; valmax(4) = max(prix[…], prix[…] + …, prix[…] + …, prix[…] + …) = ….

Pour les longueurs 1, 2 et 3, on applique le cas d'arrêt. Pour 4 m, on compare la vente entière aux trois choix possibles de première coupe, dont le morceau restant est optimisé.

> **RÉPONSE À RETENIR**
>
> - `valmax(1) = prix[1] = 1`
> - `valmax(2) = prix[2] = 5`
> - `valmax(3) = prix[3] = 8`
> - `valmax(4) = max(prix[4], prix[3] + valmax(1), prix[2] + valmax(2), prix[1] + valmax(3)) = max(9, 8 + 1, 5 + 5, 1 + 8) = 10`

Question 10.

Sujet p. 8

> Expliquer pourquoi, si n ≥ 4, on a bien : valmax(n) = max(prix[n], prix[n-1] + valmax(1), prix[n-2] + valmax(2), …,

FormaV - document d'entraînement non officiel

prix[1] + valmax(n-1)).

Pour une planche de longueur `n`, on peut soit la vendre entière, ce qui rapporte `prix[n]`, soit la découper. Dans ce dernier cas, on choisit un premier morceau de longueur `k`, avec $1 \leq k \leq n - 1$. Ce morceau rapporte `prix[k]`, et le reste, de longueur $n - k$, rapporte au maximum `valmax(n - k)`. En examinant tous les choix de `k`, on compare toutes les premières coupes possibles avec la vente entière.

RÉPONSE À RETENIR

On obtient donc bien `valmax(n) = max(prix[n], prix[n-1] + valmax(1), prix[n-2] + valmax(2), …, prix[1] + valmax(n-1))` : les termes correspondent à toutes les longueurs possibles du premier morceau, puis au meilleur résultat pour le reste.

Question 11.

Sujet p. 8

Recopier et compléter les lignes 2 à 7 du code de la fonction valmax, qui prend en paramètre un entier n et renvoie la valeur maximale obtenue pour une planche de longueur n selon les prix de la variable globale prix.

Si $n \leq 3$, on applique le cas d'arrêt. Sinon, on initialise le maximum à la valeur de la planche entière, puis on examine les premières coupes de longueurs 1 à $n - 1$. Pour chaque longueur, on ajoute au prix du premier morceau la valeur maximale du reste.

RÉPONSE À RETENIR

```
def valmax(n):
    if n <= 3:
        return prix[n]
    else:
        maximum = prix[n]
        for longueur in range(1, n):
            maximum = max(maximum, prix[longueur] + valmax(n - longueur))
        return maximum
```

Question 12.

Sujet p. 8

Expliquer pourquoi l'appel de valmax(7) effectuera deux appels de valmax(5).

Dans la boucle de `valmax(7)`, le choix d'une première longueur égale à 2 appelle directement `valmax(5)`. Le choix d'une première longueur égale à 1 appelle `valmax(6)` ; dans le calcul de cet appel, une première longueur égale à 1 appelle à son tour `valmax(5)`. Ces deux appels à `valmax(5)` sont distincts : cette fonction ne mémorise pas les résultats.

RÉPONSE À RETENIR

L'appel de `valmax(7)` effectue deux appels à `valmax(5)` : l'un directement, et l'autre au cours du calcul de `valmax(6)`.

C

Question 13.

Sujet p. 8, 9

> Recopier et compléter les lignes 2 à 8 du code de la fonction valmax_dynamique. Si memo[n] ne vaut pas -1, la valeur maximale a déjà été mémorisée et memo[n] vaut cette valeur maximale : il faut alors la renvoyer. Sinon, il faut calculer la valeur maximale en utilisant les appels récursifs et affecter à memo[n] la valeur maximale obtenue.

Si la valeur de `memo[n]` est différente de `-1`, elle est déjà calculée et on la renvoie. Sinon, on calcule le maximum comme précédemment, mais en utilisant les appels à `valmax_dynamique`, puis on mémorise le résultat dans `memo[n]`.

RÉPONSE À RETENIR

```
def valmax_dynamique(n):
    if memo[n] != -1:
        return memo[n]
    else:
        maximum = prix[n]
        for longueur in range(1, n):
            maximum = max(maximum, prix[longueur] + valmax_dynamique(n - longueur))
        memo[n] = maximum
    return maximum
```

Question 14.

Sujet p. 8, 9

> Donner en justifiant le nombre d'appels récursifs effectués par l'appel de valmax_dynamique(6), après l'exécution de valmax_dynamique(5).

Après l'appel de `valmax_dynamique(5)`, les valeurs pour les longueurs 1 à 5 sont mémorisées : les longueurs 1, 2 et 3 l'étaient dès l'initialisation, et les calculs ont complété notamment `memo[4] = 10` et `memo[5] = 13`. Pour calculer `valmax_dynamique(6)`, la boucle appelle la fonction pour les longueurs 5, 4, 3, 2 et 1. Chacun de ces appels renvoie aussitôt une valeur mémorisée : ils ne provoquent donc aucun autre appel récursif.

RÉPONSE À RETENIR

L'appel de `valmax_dynamique(6)` effectue **5 appels récursifs** : `valmax_dynamique(5)`, `valmax_dynamique(4)`, `valmax_dynamique(3)`, `valmax_dynamique(2)` et `valmax_dynamique(1)`. Aucun ne déclenche d'appel supplémentaire.

Question 15.

Sujet p. 9

> Expliquer ce qui se serait passé si la variable memo avait été initialisée avec uniquement des valeurs égales à -1.

Avec toutes les cases initialisées à `-1`, les premières valeurs, y compris celles pour les longueurs 1, 2 et 3, ne seraient pas disponibles d'emblée. Elles seraient calculées par les appels récursifs puis mémorisées. Le calcul se termine, car chaque appel porte sur une longueur strictement plus petite ; pour `n = 1`, la boucle est vide. Les résultats restent donc corrects, mais des calculs supplémentaires sont nécessaires au début.

RÉPONSE À RETENIR

Le calcul ne serait ni incorrect ni infini : les valeurs seraient calculées récursivement puis mémorisées. Les valeurs de `valmax(1)`, `valmax(2)` et `valmax(3)` devraient toutefois être calculées au lieu d'être disponibles dès l'initialisation.

Question 16.

Sujet p. 7, 9

Écrire une fonction valeur_max_stock qui prend en paramètre un dictionnaire effectif et qui renvoie la valeur maximale, en euro, que peut rapporter le stock à l'entreprise en optimisant le découpage de chaque planche afin qu'elle rapporte le maximum. La variable memo vient d'être initialisée comme indiqué dans le sujet.

Pour chaque longueur présente dans le stock, on multiplie le nombre de planches par la valeur maximale qu'une planche de cette longueur peut rapporter. On additionne ensuite ces montants. La fonction `valmax_dynamique` réutilise les valeurs mémorisées dans `memo`. Si le dictionnaire est vide, la boucle ne s'exécute pas et la fonction renvoie 0.

RÉPONSE À RETENIR

```
def valeur_max_stock(effectif):
    valeur_totale = 0
    for longueur in effectif:
        valeur_totale = valeur_totale + effectif[longueur] *
valmax_dynamique(longueur)
    return valeur_totale
```

Sujet : https://aeif.fr/wp-content/uploads/2026/06/Antilles-Guyane_juin_2026_sujet_2.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.