Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour consulter tous nos sujets d'annales et corrigés du Bac Général Maths + NSI, veuillez visiter :
bac-maths-nsi.fr/annales-sujets-et-corriges

Amérique du sud - novembre 2025 - sujet 2 (corrigé)

Exercice 1 (Bases de données, SQL, programmation Python, listes)
Partie A

1. Cette requête renvoie la table suivante :

nom
NOUMEA

2. La requête suivante convient :

```
SELECT nom FROM station ORDER BY nom
```

3. La requête suivante convient :

```
SELECT forceVent, dirVent FROM observation
JOIN station
ON observation.idStat = station.idStat
WHERE nom = 'BOURAKE' AND date = '2023010214'
```

4. La requête suivante convient :

```
SELECT COUNT(idObs) FROM observation
```

5. Le schéma relationnel de la table `meteo` en supprimant les données `hauteur`, `precip`, `forceVent` et `dirVent` est :

`meteo(idStat : INT, nom : TEXT, latitude : REAL, longitude : REAL, idObs : INT, date : DATE)`

Partie B

6. Les commandes créent une liste à partir du fichier `'observations.csv'`, suppriment les noms des champs, puis transforment les chaînes de caractères en entiers ou en flottants pour certains champs ; le résultat obtenu est une version correctement typée du premier enregistrement de la table.
7. L'instruction nécessaire à l'utilisation du module Python `math` est `import math`
8. Le code suivant convient :

```
def coord(l_obs, stat_ref):
    for obs in l_obs :
        if obs[1] == stat_ref :
            return (obs[2], obs[3])
```

9. L'algorithme suivant convient :

```
fonction liste_stations(l_obs, stat_ref, dist)
    on initialise une liste vide l_ident qui contiendra la liste des identifiants
    pour chaque station stat de la liste l_obs
        si la distance entre stat et stat_ref est inférieure à dist
            appendre l'identifiant de stat à la liste l_ident
        fin si
    fin pour
    renvoyer l_ident
fin fonction
```

10. La fonction suivante convient :

```
def nettoyage(l_obs, stat_ref):
    l_stations = liste_stations(l_obs, stat_ref, 2000)
    l_temp = []
    for obs in l_obs:
        if obs[0] in l_stations:
            l_temp.append(obs[9])
    return l_temp
```

11. L'une des fonctions suivantes convient :

```
def moyenne(L):
    return sum(L)/len(L)
```

```
def moyenne(L):
    somme = 0
    nombre = 0
    for x in L:
        somme = somme + x
        nombre = nombre + 1
    return somme / nombre
```

12. Les commandes suivantes conviennent :

```
>>> liste_obs = creation_liste_obs('observations.csv')
>>> liste_obs = supp_champs(liste_obs)
>>> transtype(liste_obs)
>>> L = [obs[9] for obs in liste_obs if obs[5] // 100 == 20240101 and
distance('Paris_11', obs[0]) <= 2000]
>>> moyenne(L)
```

Exercice 2 (Structure de pile, POO et algorithmique)

1. On peut terminer le jeu en versant le tube 4 dans le tube 3 en partant de la situation de la figure 4.

Partie A : les tubes

2. La structure de pile est une structure linéaire de type LIFO (*Last-In First-Out*), dont les méthodes `empiler` et `depiler` permettent d'ajouter un élément au sommet et de récupérer le sommet en le supprimant de la pile, respectivement.
3. Les lignes 11 et 12 du code de la classe `tube` permettent d'empiler la couleur et de mettre à jour la prochaine position où on pourra empiler une autre couleur.
4. Le code suivant convient :

```
def depiler(self):
    if self.taille > 0:
        self.taille = self.taille - 1
        couleur = self.contenu[self.taille]
        self.contenu[self.taille] = 0
        return couleur
    else:
        return -1
```

5. La méthode suivante convient :

```
def est_plein(self):
    return self.taille == 3
```

6. La méthode suivante convient :

```
def est_homogene(self):
    if self.taille < 2:
       return True
    if self.contenu[1] != self.contenu[0]:
        return False
    if self.taille == 2:
       return True
    return self.contenu[2] == self.contenu[1]
```

7. La méthode suivante convient :

```
def derniere_couleur(self):
    if self.taille == 0:
       return -1
    return self.contenu[self.taille - 1]
```

8. Le code suivant convient :

```
def verser(self, other):
    while not self.est_vide() and (other.est_vide() or other.taille<3 \
          and self.derniere_couleur() == other.derniere_couleur()) :
        couleur = self.depiler()
        other.empiler(couleur)
```

Partie B : le jeu

9. L'instruction `tube2.verser(tube1)` permet de faire passer la variable `etat` de la représentation en figure 2 à celle de la représentation en figure 3.
10. La fonction suivante convient :

```
def gagne(etat):
    for pile in etat:
        if not etat.est_homogene():
            return False
    return True
```

Exercice 3 (POO, graphes et réseaux)

Partie A

1. La valeur associée à la clé `1` dans ce dictionnaire est `[2, 5]`.
2. La fonction suivante convient :

```
def voisins(graphe, k):
    return graphe[k]
```

3. La fonction suivante convient :

```
def degre_du_sommet(graphe, sommet):
    return len(graphe[sommet])
```

4. La fonction suivante convient :

```
def degre_sommets(graphe):
    return [(sommet, degre_du_sommet(graphe, sommet)) for sommet in graphe]
```

5. A la ligne 4, la boucle `for` fait varier `i` de `0` à `len(l_deg)` **inclus** mais les éléments de `l_deg` sont indexés de `0` à `len(l_deg)` **exclus**, ce qui explique l'erreur d'index déclenché à la ligne 6 lorsque `i` vaut 3 dans le cas d'usage testé. Il suffit de supprimer le `+1` pour corriger l'erreur.
6. Le tri de `tri_liste` est un tri par sélection.
7. La fonction suivante convient :

```
def tri_sommets(graphe):
    liste_couples = tri_liste(degre_sommets(graphe))
    return [t[0] for t in liste_couples]
```

8. La variable `coloration_sommets` est un dictionnaire qui, après exécution de la boucle des lignes 7 et 8 vaut :

```
{1: None, 2: None, 3: None, 4: None, 5: None, 6: None, 7: None,8: None, 9: None}
```

9. La fonction fournit le coloriage suivant :

```
{1: 'Vert', 2: 'Bleu', 3: 'Rouge', 4: 'Vert', 5: 'Rouge', 6: 'Bleu', 7: 'Bleu',
 8: 'Bleu', 9: 'Rouge'}
```

Partie B

10. La commande `cp prog1.py ../travail/TP` convient
11. La commande `ping 190.12.10.25` convient
12. Une adresse possible pour l'ordinateur P2 est `12.128.42.42`
13. Le chemin emprunté par un paquet de données allant de l'ordinateur P1 à l'ordinateur P2 est P1-S1-R1-R2-R3-R8-R9-S2-P2
14. Le protocole de routage qui semble être utilisé est RIP, puisque avec OSPF R1-R5-R6-R4-R3 serait plus court que R1-R3.
15. Les coûts pour des liaisons de 100 Mbits/s, 1 Gbits/s et 10 Gbits/s sont respectivement $10^8/(100 \times 10^6) = 1$, $10^8/10^9 = 0{,}1$ et $10^8/(10 \times 10^9) = 0{,}01$
16. La route qui sera empruntée par le paquet de données envoyé de l'ordinateur P1 à l'ordinateur P2, en respectant le protocole OSPF, sera donc P1-S1-R1-R5-R6-R4-R3-R8-R9-S2-P2

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.