l'Etudiant

BAC 2026
Correction épreuve de NSI
Mardi 17 juin 2026

Exercice 1

1. L'adresse IP du réseau est 172.16.0.0 (16 derniers bits à 0).

2. L'adresse de diffusion est 172.16.255.255 (16 derniers bits à 1).

3. On peut y mettre 2^{16} -2 machines soit 65534. (Non demandé lors du barème car calculatrice non autorisée)

4.

Destination	passe par	Nb de sauts
R2	R3	2
R3	R3	1
R4	R4	1
R5	R4	2

5. Pour aller de PCA01 à PCB01, un paquet passe par R1, R3, R2.

6. Si R3 tombe en panne, les paquets passent alors par :
R1 → R4 → R5→ R2

7.

Type	Débit (bit/s)	Coût
Ethernet	10^7	10
Fast Ethernet	10^8	1
Fibre	10^9	0 ,1

8. En OSPF, on minimise les coûts. Donc pour aller de R1 à R4, les paquets transiteront ainsi : R1 → R3 puis R3 → R4 soit un coût de 1,1 contre 10.

9. Il n'y a aucune structure conditionnelle, et on utilise la méthode append directement : MAX_ROUTES n'est jamais utilisée. Le code n'en tiendra pas compte.

10.

```
self.capacite = capacite #on crée l'attribut self.capacite à partir de la valeur fournie

def ajouter (self, route) :
    if len(self.routes)<self.capacite :
        self.routes.append(route)
    else :
        print('capacité maximale atteinte')
```

11.

```
def afficher(self):
    for route in self.routes :  #on parcourt les routes
        for routeur in route : # on parcourt chaque routeur dans chaque route
            print (routeur)
        print('---') #on a fini une route, on affiche la séparation
```

Exercice 2

PARTIE A

1.

>>>melange[2][1]

10 #3e ligne, 2e item

2. valeurs = [k for k in range (1,17)]

3. shuffle fonctionnant « en place », il suffit de l'appeler, sans réaffecter valeurs

>>>shuffle(valeurs)

4. assert len(valeurs)==n*n, "la liste n'a pas la bonne longueur"

5. il faut échanger les indices pour que le résultat soit correct.

grille[i][j] = valeurs[i * n + j]

PARTIE B

6. la grille [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 16, 15, 14]]

possède 2 inversion car 16 < 15 et 16 < 14.

la distance est de 2 (séparée de 2 colonnes)

Donc la grille est résoluble.

7. Une proposition de test :

assert compte_inversions([1, 4, 2, 3]) == 2

8. Le problème est que dans la 2e boucle for, on reparcourt toute la liste depuis le début. Or, on doit la parcourir à partir de l'indice i (voire i+1 si on veut optimiser une comparaison) :

```
  for j in range(len(valeurs)):
```

devient

```
  for j in range(i,len(valeurs)):
```

9.

```
def distance_tuile_vide(grille, n):
    num_tuile_vide = n * n
    for i in range(n):
        for j in range(n):
            if grille[i][j] == n*n: #on vérifie si on est sur la tuile vide
                return (n-1)-i + (n-1)-j #on renvoie la somme des lignes et colonne pour aller au bout
            #n-1 est le numéro de ligne ou colonne final et i et j les coordonnées de la tuile vide
```

10.

```
def est_resoluble(valeurs, n):
    grille = en_grille(valeurs, n)
    inv = compte_inversions(valeurs) #attention, valeurs en ligne attendue
    dis = distance_tuile_vide(grille,n) #attention, grille attendue
    return (inv + dis) % 2 == 0 #renvoie True si la somme est paire, False sinon.
```

PARTIE C

11.
graphe_4_4[9] à 4 voisins donc voici une réponse possible : [5,8,13,10]

12.

```
def melange_graphe(nb_dep, n, graphe):
    valeurs = [i for i in range (n*n)]#liste aplatie rangée
    num_tuile_vide = n * n
    actuelle = num_tuile_vide - 1#position de la tuile vide
    for k in range(nb_dep):
        prochaine = choice(graphe[actuelle]) #on tire au hasard une case d'un déplacement
        ##difficile de faire un échange sans 1 ligne supplémentaire
        temp = valeurs[actuelle]
        valeurs[actuelle] = valeurs[prochaine]
        valeurs[prochaine] = temp
        actuelle = prochaine
    return en_grille(valeurs, n)
```

Exercice 3

PARTIE A

1. Un couple nom, prénom peut ne pas être unique : il ne peut être une clé primaire.

2. Les deux attributs pointent vers les clés étrangères id_utilisateur et id_trajet qui sont des nombres entiers donc ce sont des nombres entiers.

3. SELECT COUNT(*) FROM inscription WHERE trajet = 1291;

4. SELECT id_trajet
FROM trajet
WHERE arrivee = 'Nancy' AND heure = '2026-06-19' ORDER BY depart ;

l'Etudiant

5. INSERT INTO trajet (id_trajet, depart, arrivee, heure, nb_places, conducteur) VALUES (1295, 'Nancy', 'Allain', '2026-06-19 17:45:00', 3, 25);

6. UPDATE trajet SET heure = '2026-06-19 18:40:00' WHERE id_trajet = 1294;

7. On ne peut effacer ce trajet car il est lié à des inscriptions. Il faudrait supprimer ou rediriger les inscriptions pour pouvoir l'effacer. Ceci est lié à la contrainte sur l'existence d'une clé étrangère.

8. SELECT DISTINCT nom, prenom FROM utilisateur
JOIN inscription ON utilisateur.id_utilisateur = inscription.passager
JOIN trajet ON inscription.trajet = trajet.id_trajet
WHERE trajet.conducteur = 25;

9.
graphe = { "C": ["P1", "P2"],
"P1": ["C", "A"],
"A": ["P1", "P2", "P3"],
"P2": ["C", "A", "M"],
"P3": ["A", "M"],
"V": ["M"],
"M": ["P2", "P3", "V"] }

10. Une file est une structure de données de type FIFO (First In, First Out) qui organise les entrées et sorties des données ainsi :
On récupère les données par ordre d'entrée.

11. On démarre de P1 et on parcourt le graphe en largeur : (utilisation d'une file)

P1, A, C, P2, P3, M , V

12. Quand on utilise une file, on parcourt les voisins d'un nœud, puis on passe au nœud de même distance et on parcourt ses voisins etc. C'est un parcours en largeur.

13.

```
def excentricite(graphe, sommet):
    """ graphe : dictionnaire
    {sommet : liste de sommets adjacents }
    renvoie l'excentricité de sommet
    dans le graphe (int) """
    dico = dico_distance(graphe, sommet)
    distance_max = 0
    for s in dico: #on parcourt les sommets du graphes
        if dico[s] > exc: #on récupère la distance maximale
            distance_max = dico[s]
    return distance_max
```

14. Si on calcule les excentricités entre les points de RDV, on trouve :
P1 a pour excentricité 4
P2 a pour excentricité 2
P3 a pour excentricité 3

P2 pourrait être le meilleur point de RDV s'il faut minimiser la distance maximale à parcourir. (Et si on le fait avec les moyennes des excentricités, le point P2 est aussi le meilleur, mais c'est plus compliqué à faire sans calculette.)