Exercice 1 : Cœur de réseau
Partie A

1. **Attention :** La métrique est le nombre de routeur traversés (et non le nombre de saut).

Table de routage R_1		
Destination	Prochain saut	Distance
R_2	R_2	0
R_3	R_2	1
R_4	R_4	0
R_5	R_5	0
R_6	R_5	1

2. La requête d'un utilisateur du LAN1 vers INTERNET utilise selon le protocole RIP le chemin :
LAN1 - R1 - R5 - R6 – INTERNET

3.

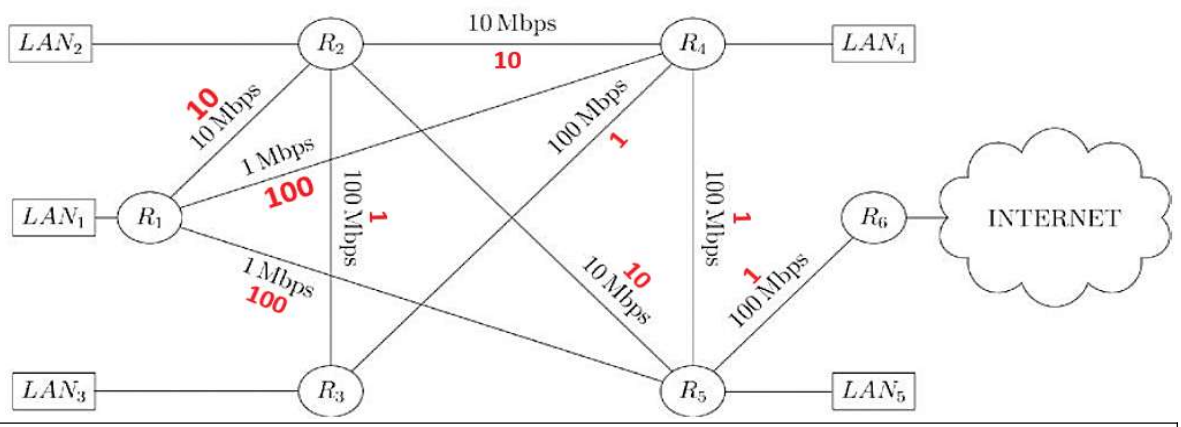


Table de routage R_1		
Destination	Prochain saut	Distance
R_2	R_2	10
R_3	R_2	11
R_4	R_2	12
R_5	R_2	13
R_6	R_2	14

4. La requête d'un utilisateur du LAN1 vers INTERNET utilise selon le protocole OSPF le chemin :
LAN1 – R1 - R2 - R3 – R4 - R5 – R6 – INTERNET ayant un coût/distance de 14.

5. Après une panne du routeur 2, la requête d'un utilisateur du LAN1 vers INTERNET utilise selon le protocole OSPF le chemin :
LAN1 – R1 - R5 – R6 – INTERNET ayant un coût/distance de 101.

Partie B

6. Détails des calculs : $1_{(10)} = 00000001_{(2)}$ et $100(10) = 64 + 32 + 4 = 01100100_{(2)}$

Calcul d'une adresse de réseau				
machine (déc. pointée)	192	168	1	100
machine (binaire)	11000000	10101000	**00000001**	**01100100**
masque (binaire)	11111111	11000000	00000000	00000000
réseau (binaire)	11000000	**10000000**	**00000000**	**00000000**
réseau (déc. pointée)	192	**128**	**0**	**0**

7. Adresse de Braodcast.
Détails des calculs : $10111111_{(2)} = 128+32+16+8+4+2+1$ ou $255-64 = 191_{(10)}$

Calcul d'une adresse de réseau				
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 (déc. pointée)	**192**	**191**	**255**	**255**

8.
Adresse du réseau LAN_1 : 172.16.0.0/16
Adresse de broadcast : 172.16.255.255/16
Nombre total d'adresses machines disponible : 2^{16} – 2 (reseau et broadcast) = 65534

9.

```
13      for i in range(4):
14          # Opération booléenne :
15          tmp.append(ip[i] & crible[i])
16      return ".".join([str(element) for element in tmp])
```

ou

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

Attention : Il faut convertir les entiers de la liste **tmp** en chaînes de caractères pour éviter l'erreur : **TypeError: sequence item 0: expected str instance, int found** de la méthode **join**.

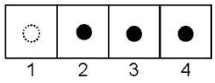
10.

```
17    for index in range(4):
18        somme = liste_courante[3 - index] + retenue
19        valeur, retenue = somme%256, somme//256
20        liste_suivante = [str(valeur)] + liste_suivante
21    return '.'.join(liste_suivante)
```

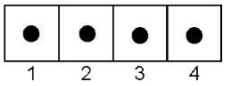
Explication : On ajoute 1 au dernier octet (4e octet à droite). Si cet octet est égal à 256, on ramène la valeur à 0 (somme%256) et on retient 1 (somme//256) pour l'ajouter à l'octet suivant (3e octet).

Exercice 2 : Jeu du baguenaudier

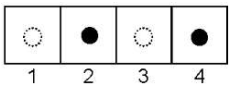
1.



Il est possible de jouer la case 1 :



ou la case 3 :



2.

```
def initialiser(n):
    tab = []
    for i in range(n):
        tab.append(False)
    return tab
```

Autre solution :

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

3.

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

4.

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

5.

```
def coup_valide(tab, case):
    if case==0:
        return True
    elif case>=0 and case<len(tab) and case==indice_premiere_case_occupee(tab)+1:
        return True
    else:
        return False
```

6.

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

7. Principe : Pour remplir un baguenaudier de taille n, il faut partir d'un baguenaudier rempli de taille n-1 puis le vider jusqu'à la taille n-2 afin que seule la dernière case n-1 soit remplie et que l'on puisse remplir la case n.

```
def vider(n):
    if n == 1:
        print('Vider case 1')
    elif n > 1:
        vider(n-2)
        print(f'Vider case {n}')
        remplir(n-2)
        vider(n-1)
```

8.

La pile d'appel récursif est :

```
vider(3)
    vider(1)
        "Vider case 1"
    "Vider case 3"
    remplir(1)
        "Remplir case 1"
    vider(2)
        vider(0)
        "Vider case 2"
        remplir(0)
        vider(1)
            "Vider case 1"
```

L'affichage obtenu en console est :

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

9.

```
def remplir(n):
    if n == 1:
        print('Remplir case 1')
    elif n > 1:
        remplir(n-1)
        vider(n-2)
        print(f'Remplir case {n}')
        remplir(n-2)
```

10.

Pour un baguenaudier de 2000 cases, le nombre d'appels récursifs va explosé et dépasser la limite de la pile d'appel de 1000. Une erreur de type `RecursionError` arrêtera alors le programme.

Exercice 3 : Gestionnaire de mots de passe

Partie A

1. La CNIL recommande d'avoir des mots de passe d'au moins 12 caractères :

```
gen_mdp(12, True, True, False)
```

2. D'après la table ASCII, les minuscules vont de 97 (a) à 122 (z), les majuscules de 65 (A) à 90 (Z) et les caractères spéciaux de 33 à 47 et de 58 à 64 :

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

3.

```
11  jeu_caracteres = []
12  if cont_min:
13      jeu_caracteres += minuscules
14  if cont_maj:
15      jeu_caracteres += majuscules
16  if cont_spe:
17      jeu_caracteres += caracteres_speciaux
```

4.

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

5. Les caractères composants le mot de passe sont piochés au hasard dans une liste de majuscules, minuscules et caractères spéciaux mais rien ne vérifie qu'il y a bien une minuscule et un caractère spécial qui ont été sélectionnés.

6. L'attribut `mot_de_passe` étant définit comme clé primaire de la table `compte`, la contrainte de relation impose des valeurs différentes de cet attribut pour chaque enregistrement.

Partie B

7.

```
SELECT url FROM site;
```

8.

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

9.

```
SELECT id_site
FROM compte
WHERE renouvellemement < '2024-03-20';
```

10. Le format AAAA-MM-JJ permet d'assurer que l'ordre lexicographique des dates est équivalent à l'ordre des dates elle-même.
En effet **2025-03-21** est bien supérieur à **2025-02-23**
alors que **21-03-2025** est inférieur à **23-02-2025**.

11.
```sql
SELECT c.utilisateur, c.mot_de_passe
FROM compte AS c
JOIN site AS s ON c.id_site = s.id
WHERE s.nom_site = 'Votremailp'
ORDER BY c.renouvellement;
```

12. La solution à deux table a un avantage dans le cas où Alice possède plusieurs comptes sur un même site car cela évite la redondances des informations des sites concernés.

Partie C

13.
```
chiffrement('gestionnaire.db', '../Perso/secret.db', '../Perso/cle')
```

14. Rappel : Un caractère hexadécimal correspond à 4 caractères binaires : $A_{(16)} = 1010_{(2)}$.
$A3_{(16)}$ = $10100011_{(2)}$
$59_{(16)}$ = $01011001_{(2)}$
XOR $11111010_{(2)} = FA_{(16)}$

15. Pour démontrer **(a XOR b) XOR b = a**, il faut établir la table de vérité :

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

16. Ce chiffrement est symétrique car la clé de déchiffrement est la même que celle de chiffrement.

17. Tous les utilisateurs `-rw-r--r--` possèdent le droit de lecture sur le fichier secret.db et peuvent donc notamment l'ouvrir et tenter une attaque.
Alice peut corriger ce problème en supprimant ce droit à tous les utilisateurs.

Partie D

18.

P1 : Alice a choisi `mot_de_passe` comme clé primaire de la table `compte`, ce qui impose l'unicité des mots de passe pour chaque site.
P2 : L'outil d'Alice lui permet de générer des mots de passe suffisamment long même si cela n'est pas imposé.
P3 : Alice a décidé de chiffrer sa base de donnée de mot de passe, ce qui ajoute un niveau de sécurité mais l'humain reste toujours le maillon faible. Alice doit s'appliquer cette règle et ne rien communiquer à d'autres concernant son système de mot de passe.
P4 : Alice a développé son projet dans le but de réaliser cette consigne : créer un gestionnaire de mot de passe.