```
curseur.close()
connection.close()
```

8. Écrire un script Python permettant de créer le dictionnaire `dict_emprunts` qui, à chaque jeu emprunté, associe le nombre de fois où il a été emprunté.

On veut créer un podium des jeux les plus souvent empruntés. Comme il peut y avoir des égalités à la première, deuxième ou troisième place, il peut y avoir plus de trois jeux sélectionnés sur le podium.

Par exemple, si le dictionnaire des emprunts est :

```
dict_emprunts = {
    "Terraforming Mars": 25,
    "Codenames": 22,
    "Agricola": 18,
    "Puerto Rico": 18,
    "Caylus": 18,
    "Dominion": 22,
    "Dixit": 12
}
```

il y aura sur le podium les jeux “Agricola”, “Puerto Rico” et “Caylus” puis les jeux “Dominion” et “Codenames” et enfin le jeu “Terraforming Mars”.

Pour modéliser ce podium en Python, on va utiliser une liste de trois listes.

Pour l’exemple précédent, cette liste sera :

`[["Agricola", "Puerto Rico", "Caylus"], ["Dominion", "Codenames"], ["Terraforming Mars"]]`.

9. Proposer un script Python permettant de générer ce podium.

Exercice 3 (8 points)

Cet exercice porte sur la programmation de base en Python, la sécurisation des communications et les réseaux.

Partie A - La méthode du *masque jetable*

Dans cette partie, on s’intéresse à une méthode de chiffrement dite du *masque jetable*. Voici ce que l’on peut lire sur le site Wikipédia :

Le chiffrement par la méthode du masque jetable consiste à combiner le message en clair avec une clé présentant les caractéristiques très particulières suivantes :

- *la clé doit être une suite de caractères au moins aussi longue que le message à chiffrer ;*

- *les caractères composant la clé doivent être choisis de façon totalement aléatoire ;*
- *chaque clé, ou « masque », ne doit être utilisée qu'une seule fois (d'où le nom de masque jetable).*

*Illustrons cette méthode par un exemple : on souhaite chiffrer le message **HELLO** avec la clé aléatoire, ou « masque », **WMCKL**.*

Pour cela, on attribue un nombre à chaque lettre, par exemple le rang dans l'alphabet, de 0 à 25.

Tableau de correspondance													
Lettre	A	B	C	D	E	F	G	H	I	J	K	L	M
Rang	0	1	2	3	4	5	6	7	8	9	10	11	12

Tableau de correspondance													
Lettre	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Rang	13	14	15	16	17	18	19	20	21	22	23	24	25

Ensuite, on additionne la valeur du rang de chaque lettre du message avec la valeur du rang correspondante dans le masque.

Enfin, si le résultat est supérieur à 25 on soustrait 26 (calcul dit « modulo 26 »).

*Ainsi, le chiffrement du message **HELLO** avec la clé **WMCKL** donne le message chiffré **DQNVZ** comme le montre l'illustration suivante.*

```
   7 (H)   4 (E)  11 (L)  11 (L)  14 (O) message
+ 22 (W)  12 (M)   2 (C)  10 (K)  11 (L) masque
= 29      16      13      21      25     masque + message
=  3 (D)  16 (Q)  13 (N)  21 (V)  25 (Z) masque + message  modulo 26
```

Figure 1. Exemple de chiffrement par la méthode du masque jetable

Source : d'après l'article Masque jetable de Wikipédia en français (https://fr.wikipedia.org/wiki/Masque_jetable)

Dans cet exercice, on ne travaillera que sur des chaînes de caractères écrites en majuscules non accentuées (les 26 caractères allant de 'A' à 'Z').

1. Chiffrer, par la méthode du *masque jetable*, le message **LIBRE** à l'aide de la clé **EYQMT**.

En Python, on crée une fois pour toute la variable `alphabet` qui sera accessible et utilisable dans toutes les fonctions. Celle-ci contient la liste des 26 lettres de l'alphabet rangées dans l'ordre alphabétique :

```
alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
'W', 'X', 'Y', 'Z']
```

2. Écrire une fonction Python `indice` qui prend pour paramètre une liste `L` et renvoie l'indice de `element` dans la liste `L`.

   On supposera que chaque élément de la liste `L` n'y apparaît qu'une seule fois et que `element` est bien présent dans la liste `L`.

   Par exemple, l'appel `indice(alphabet, 'K')` renvoie l'entier `10`.

3. Écrire une fonction Python `lettres_vers_indices` qui prend pour paramètre une chaîne de caractères et renvoie, dans l'ordre, la liste des indices de ces caractères dans l'alphabet.

   Par exemple, l'appel `lettres_vers_indices('HELLO')` renvoie la liste d'entiers `[7, 4, 11, 11, 14]`.

On dispose également d'une fonction `indices_vers_lettres`, qu'on ne demande pas d'écrire, permettant de convertir une liste d'entiers, compris entre 0 et 25, en une chaîne de caractères.

Par exemple, l'appel `indices_vers_lettres([3, 16, 13, 21, 25])` renvoie la chaîne de caractères `'DQNVZ'`.

Ci-après, on donne une fonction Python `chiffrement` incomplète, qui, à partir d'un message `msg` et d'une clé `cle` entrés en paramètres, renvoie la chaîne de caractères représentant le message chiffré par la méthode du *masque jetable*.

```
 1 def chiffrement(msg, cle):
 2     assert len(cle) >= len(msg), 'impossible'
 3     indices_msg = lettres_vers_indices(msg)
 4     indices_cle = lettres_vers_indices(cle)
 5     n = len(msg)
 6     indices_msg_chiffre = []
 7     for k in range(n):
 8         ind = ...
 9         if ind >= 26:
10             ind = ...
11         indices_msg_chiffre.append(ind)
12     msg_chiffre = indices_vers_lettres(...)
13     return msg_chiffre
```

4. Recopier et compléter les lignes 7 à 13 de la fonction `chiffrement`.

5. Indiquer, en justifiant, ce que l'on observe lors de l'appel `chiffrement('RESEAU', 'GFTZ')`.

On s'intéresse maintenant au déchiffrement d'un message chiffré par la méthode du *masque jetable*.

Par exemple, le déchiffrement du message **DQNVZ** avec la clé **WMCKL** donne le message **HELLO**.

6. Déchiffrer le message **GMEDH** avec la clé **FVEIT**.
7. Expliquer comment procéder pour déchiffrer un message lorsqu'on connaît la clé.

On souhaite maintenant écrire, en Python, une fonction `dechiffrement` qui permet de déchiffrer un message chiffré par la méthode du *masque jetable*.

Pour cela, on s'inspire de la fonction `chiffrement` dans laquelle les paramètres ainsi que les lignes 2 à 5 sont inchangées. On décide cependant de remplacer, ligne 6, le nom de la variable `indices_msg_chiffre` par le nom plus explicite `indices_msg_dechiffre`.

8. Adapter les lignes 6 à 13 de la fonction `chiffrement` pour obtenir la nouvelle fonction `dechiffrement`.

Partie B - Sécurisation des communications

9. Expliquer la différence entre un algorithme de chiffrement symétrique et un algorithme de chiffrement asymétrique.

Alice souhaite envoyer un message à Bob par l'intermédiaire d'un réseau informatique en utilisant un algorithme de chiffrement asymétrique.

Pour cela, Bob envoie à Alice sa clé publique. Alice chiffre ensuite le message à l'aide de la clé publique de Bob qu'elle vient de recevoir, puis elle envoie ce message chiffré à Bob.

10. Indiquer comment Bob peut déchiffrer le message que lui envoie Alice.
11. Expliquer comment une tierce personne pourrait se faire passer pour Alice sans que Bob ne s'en aperçoive.
12. Expliquer brièvement le fonctionnement du protocole HTTPS.
13. Expliquer pourquoi, pour sécuriser intégralement les communications sur Internet, on utilise le protocole HTTPS plutôt qu'un chiffrement asymétrique.

Partie C - Réseaux

Bob et Marc travaillent pour une petite compagnie d'assurances.

Leurs postes de travail font partie d'un même réseau local géré par l'administratrice système qui dispose du bloc d'adresses IPv4 192.168.110.0/24.

La notation /24 situé à la suite de l'adresse 192.168.110.0 signifie que le masque de sous-réseau du réseau de cette entreprise est 255.255.255.0 : les trois premiers octets d'une adresse IP sur ce réseau permettent donc d'identifier la partie réseau de l'adresse, alors que le dernier octet permet d'identifier la partie hôte et est propre à chaque machine sur le réseau. Ce sous-réseau permet donc d'attribuer 256 adresses IPv4 différentes.

L'administratrice choisit alors d'attribuer, en représentation décimale, l'identifiant 115 pour la partie hôte du poste de travail de Bob et l'identifiant 153 pour celui de Marc.

Depuis son poste de travail, Marc souhaite tester la communication avec celui de Bob. Pour cela, il exécute la commande `ping 192.168.100.115` et obtient l'affichage suivant :

```
--- 192.168.100.115 ping statistics ---

4 packets transmitted, 0 received, 100% packet loss, time
3060ms
```

14. Expliquer l'affichage obtenu et corriger l'erreur de Marc.

Afin d'améliorer les performances et la sécurité du réseau de l'entreprise, l'administratrice système décide de séparer le réseau local en plusieurs sous-réseaux et de les relier entre eux par des routeurs. Pour cela, elle modifie le masque de sous-réseau qui devient 11111111.11111111.11111111.11100000, donné ici en représentation binaire.

15. Donner la représentation décimale de ce masque de sous-réseau.

Pour obtenir l'adresse IPv4 du sous-réseau auquel appartient une machine, il suffit d'appliquer l'opérateur binaire **ET**, bit à bit, entre le masque de sous-réseau et l'adresse IPv4 de la machine.

Par exemple, prenons le dernier octet de l'adresse IPv4 de Bob dont la représentation binaire est 01110011 : en appliquant bit à bit l'opérateur binaire **ET** entre cet octet et l'octet correspondant dans le masque, on obtient le dernier octet de l'adresse du sous-réseau, soit 01100000.

```
   1 1 1 0 0 0 0 0 (224)
ET 0 1 1 1 0 0 1 1 (115)
-----------------
   0 1 1 0 0 0 0 0 (96)
```

Le poste de travail de Bob est donc sur le sous-réseau d'adresse 192.168.110.96.

16. Indiquer le nombre total d’adresses IPv4 pouvant être attribuées sur le sous-réseau d’adresse 192.168.110.96 sur lequel se trouve Bob.

L’administratrice système attribue maintenant l’adresse IPv4 192.168.110.134 au poste de travail de Zoé, nouvelle employée de la compagnie d’assurances.

17. Donner la représentation binaire du nombre 134.

Depuis son poste de travail, Zoé exécute les deux commandes suivantes :

- **commande n°1** : `ping 192.168.110.115` ;
- **commande n°2** : `ping 192.168.110.153`.

18. Indiquer, en justifiant, laquelle de ces deux commandes a produit l’affichage :

```
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
```