#### BACCALAURÉAT GÉNÉRAL

##### ÉPREUVE D’ENSEIGNEMENT DE SPÉCIALITÉ

###### **SESSION 2025**

#### **NUMÉRIQUE ET SCIENCES INFORMATIQUES**

###### **JOUR 1**

##### Durée de l’épreuve : **3 heures 30**

##### *L’usage de la calculatrice n’est pas autorisé.*

#####

##### Dès que ce sujet vous est remis, assurez-vous qu’il est complet.

##### Ce sujet comporte 19 pages numérotées de 1/19 à 19/19.

#####

**Le sujet est composé de trois exercices indépendants.**

**Le candidat traite les trois exercices.**

# Exercice 1 (6 points)

*Cet exercice porte sur les protocoles réseaux, l’algorithmique et la POO.*

## Partie A

Un cœur de réseau est constitué d’un maillage dans lequel les routeurs $R\_{1}$ à $R\_{6}$ sont reliés par des liaisons physiques dont le débit en Mbps (Méga-bits par seconde) est indiqué sur la figure suivante.



Figure 1. Réseau

Derrière chaque routeur $R\_{1}$ à $R\_{5}$, un switch distribue un réseau local $LAN\_{1}$ à $LAN\_{5}$.

Les protocoles utilisés sont le protocole RIP et le protocole OSPF, qui minimisent respectivement le nombre de routeurs traversés et la somme des coûts des liaisons physiques empruntées.

1. Recopier et compléter la table de routage de $R\_{1}$ sachant que le protocole de routage RIP est utilisé.
* La destination est le routeur à atteindre, le prochain saut est le premier routeur traversé et la distance est le nombre de routeurs traversés.

| Table de routage $R\_{1}$ |
| --- |
| destination | prochain saut | distance |
| $$R\_{2}$$ | $$R\_{2}$$ | 0 |
| $$R\_{3}$$ | $$R\_{2}$$ | 1 |
| $$R\_{4}$$ |  |  |
| $$R\_{5}$$ |  |  |
| $$R\_{6}$$ |  |  |

1. Un utilisateur du réseau local $LAN\_{1}$ interroge, via son navigateur web, un serveur d’Internet. Détailler, suivant le protocole RIP, la route suivie dans le maillage par la requête à destination de ce serveur.

Le coût d’une liaison est donné par la relation coût = $\frac{10^{2}}{d}$, où $d$ est le débit de la liaison en Mbps.

Exemple : le coût de la liaison entre $R\_{1}$ et $R\_{2}$ est $\frac{10^{2}}{10}=10$.

Le tableau suivant donne le coût de chacune des liaisons physiques entre $R\_{1}$ et les autres routeurs du graphe qui lui sont connectés.

| Coût des liaisons depuis $R\_{1}$ |
| --- |
| routeur | $$R\_{2}$$ | $$R\_{4}$$ | $$R\_{5}$$ |
| coût | 10 | 100 | 100 |

1. Recopier et compléter la table de routage de $R\_{1}$ sachant que le protocole de routage OSPF est utilisé.
* La distance est la somme totale des coûts des liaisons physiques empruntées pour atteindre la destination.

| Table de routage $R\_{1}$ |
| --- |
| destination | prochain saut | distance |
| $$R\_{2}$$ | $$R\_{2}$$ | 10 |
| $$R\_{3}$$ |  |  |
| $$R\_{4}$$ |  |  |
| $$R\_{5}$$ |  |  |
| $$R\_{6}$$ |  |  |

Un utilisateur du réseau local $LAN\_{1}$ interroge, via son navigateur web, un serveur d’Internet.

1. Donner, selon le protocole OSPF, la route suivie dans le maillage par la requête à destination de ce serveur.

Le protocole de routage OSPF est toujours utilisé et le routeur $R\_{2}$ tombe en panne.

1. Donner la nouvelle route suivie dans le maillage par une requête partant du réseau $LAN\_{1}$ à destination d’Internet et en donner la nouvelle valeur de la distance.

## Partie B

Le réseau $LAN\_{1}$ est supposé disposer d’un serveur DHCP (Dynamic Host Control Protocol) gérant l’attribution d’adresses IPv4. Il est rappelé, ici, qu’une adresse IPv4 est une séquence de 4 octets, généralement représentée par les 4 entiers correspondants en écriture décimale, séparés par des points (notation décimale pointée). Un réseau est caractérisé par des machines dont les adresses ont en commun les $n$ mêmes premiers bits. La notation CIDR du réseau a la forme “adresse réseau / $n$”. Appliqué à une adresse du réseau, le masque permet de calculer le préfixe réseau commun. Il est constitué de 4 octets consécutifs, dont (de gauche à droite) les $n$ premiers bits sont égaux à 1 et les suivants à 0. On obtient l’adresse du réseau en effectuant un ET logique, bit à bit, entre chaque octet composant l’adresse d’une machine et l’octet qui lui correspond dans le masque.

Exemple de ET :

 l'entier 192 s'écrit : 1 1 0 0 0 0 0 0
 l'entier 255 s'écrit : 1 1 1 1 1 1 1 1
---------------------------------------------------------
ET logique, bit à bit : 1 1 0 0 0 0 0 0 ( soit l'entier 192 )

1. Recopier et compléter le tableau suivant correspondant à une machine d’un réseau d’adresse 192.168.1.100 et de masque 255.192.0.0. Ce tableau détermine l’adresse du réseau (en binaire puis en décimale pointée).

| Calcul d’une adresse de réseau |
| --- |
| machine (binaire) | 11000000 | 10101000 |  |  |
| masque (binaire) | 11111111 | 11000000 | 00000000 | 00000000 |
| réseau (binaire) | 11000000 |  |  |  |
| réseau (déc. pointée) |  192 |  |  |  |

On obtient le complémentaire d’un octet en échangeant respectivement chacun des 0 et des 1 qui le composent, par un 1 ou un 0.

Par exemple, le complémentaire de 1 1 0 0 1 0 1 0 est 0 0 1 1 0 1 0 1.

Par un procédé analogue à celui de la question précédente, l’adresse de broadcast d’un réseau est obtenue en effectuant un OU logique bit à bit entre chaque octet composant l’adresse réseau et le complémentaire de l’octet correspondant dans l’écriture binaire du masque.

Exemple de OU :

 12 s'écrit : 0 0 0 0 1 1 0 0
 9 s'écrit : 0 0 0 0 1 0 0 1
------------------------------------------------------------
OU logique, bit à bit : 0 0 0 0 1 1 0 1 ( soit l'entier 13 )

1. Recopier et compléter le tableau ci-après pour déterminer l’adresse de broadcast du réseau suivant (en binaire puis en décimale pointée).

| Calcul d’une adresse de broadcast |
| --- |
| réseau (binaire) | 11000000 | 10000000 | 00000000 | 00000000 |
| masque (binaire) | 11111111 | 11000000 | 00000000 | 00000000 |
| complément du masque (binaire) |  |  |  |  |
| broadcast (binaire) |  |  |  |  |
| broadcast (déc. pointée) |  |  |  |  |

Une machine du réseau $LAN\_{1}$ a reçu du serveur DHCP l’adresse 172.16.1.100, avec le masque 255.255.0.0.

Sa passerelle par défaut a pour adresse 172.16.255.254.

1. En déduire les informations suivantes :
	* l’adresse du réseau $LAN\_{1}$ (en décimale pointée) ;
	* l’adresse de broadcast (en décimale pointée) ;
	* le nombre total d’adresses pouvant être distribuées par le serveur, en ne tenant pas compte des éventuelles restrictions possiblement posées par l’administrateur du réseau.

On souhaite, désormais, simuler en Python le fonctionnement du réseau $LAN\_{1}$.

On donne, ci-après, les documentations des méthodes split et join.

* la documentation de la méthode split de la classe str est la suivante :
* split(self, séparateur)
 Renvoie la liste des sous-chaines de caractères délimitées par le séparateur dans l'instance courante de chaîne de caractères.
 Exemple :
>>> 'ab-pq-rs'.split('-')
['ab', 'pq', 'rs']
* la documentation de la méthode join de l’objet str :
* join(self, iterable)
Fusionne les chaînes de caractères contenues dans iterable en le liant par la sous-chaîne depuis laquelle cette méthode est appelée.
Exemple:
>>>'.'.join(['ab', 'pq', 'rs'])
'ab.pq.rs'

On commence par créer la classe IPv4 suivante.

 1 class IPv4(object):
 2 def \_\_init\_\_(self, adresse:str):
 3 """
 4 Constructeur de la classe de calcul sur une
 5 adresse IPv4 dont la notation décimale pointée
 6 est passée en paramètre.
 7 """
 8 self.adresse = adresse
 9
10 def octets(self)->list[int]:
11 """
12 Découpe l'adresse décimale pointée en la liste
13 des 4 entiers correspondants.
14 """
15 return [int(i) for i in self.adresse.split(".")]

Pour mémoire, l’opérateur & entre deux entiers effectue le ET logique bit à bit de la représentation binaire de ces entiers et renvoie l’entier correspondant à la représentation binaire ainsi obtenue.

Exemple :

>>> 192 & 255
192

1. Recopier sur la copie et compléter les lignes 15 et 16 du code de la méthode masquer de la classe IPv4, donné ci-dessous. La méthode doit correspondre au docstring.
* 1 def masquer(self, masque: str)->str:
 2 """
 3 Détermine le préfixe masqué de l'adresse,
 4 le masque (décimal pointé) étant passé en
 5 paramètre.
 6 >>> add = IPv4('192.168.1.100')
 7 >>> add.masquer('255.192.0.0')
 8 '192.128.0.0'
 9 """
10 tmp = []
11 ip = self.octets()
12 crible = IPv4(masque).octets()
13 for i in range(4):
14 # Opération booléenne :
15 tmp.append(... & ...)
16 return ".".join(...)
1. Recopier sur la copie et compléter les lignes 19 et 20 du code de la méthode adresse\_suivante de la classe IPv4, donné ci-dessous. La méthode doit correspondre au docstring.
* 1 def adresse\_suivante(self, adresse\_max:str)->str:
 2 """
 3 Détermine l'adresse décimale pointée suivant
 4 immédiatement l'adresse courante, sous réserve
 5 d'existence d'une adresse disponible
 6 >>> add = IPv4('192.168.1.100')
 7 >>> add.adresse\_suivante('192.168.1.254')
 8 '192.168.1.101'
 9 >>> add = IPv4('192.168.1.255')
10 >>> add.adresse\_suivante('192.168.255.254')
11 '192.168.2.0'
12 """
13 assert self.adresse < adresse\_max
14 liste\_courante = self.octets()
15 liste\_suivante = list()
16 retenue = 1
17 for index in range(4):
18 somme = liste\_courante[3 - index] + retenue
19 valeur, retenue = ..., ...
20 liste\_suivante = ...
21 return '.'.join(liste\_suivante)

# Exercice 2 (6 points)

*Cet exercice porte sur la programmation Python et la récursivité.*

Le jeu du baguenaudier est un jeu de casse-tête constitué d’une réglette comportant n cases, numérotées de 1 à n.

Chaque case peut être soit vide, soit contenir un pion.

« Jouer » une case consiste à placer un pion dans la case (remplir) si elle est vide ou enlever un pion (vider) si elle est remplie.

Initialement, toutes les cases sont vides.

Le but du jeu est de remplir toutes les cases du baguenaudier en suivant les règles suivantes :

* on ne peut jouer qu’une case à la fois ;
* chaque case ne peut contenir qu’un pion ;
* on peut toujours jouer la case 1 ;
* si le baguenaudier n’est ni vide ni rempli, on peut aussi jouer la case qui suit la première case remplie ;
* aucune autre case ne peut être jouée.

Exemple de situation avec un baguenaudier de 5 cases.



Figure 1. Situation baguenaudier 5 cases

Dans cette situation, on peut poser un pion dans la case 1 ou enlever le pion de la case 4 mais on ne peut pas jouer les cases 2, 3 et 5.

1. On considère un baguenaudier à 4 cases sur lequel les trois dernières cases sont remplies.
* Désigner les cases que l’on peut jouer et dessiner l’état du baguenaudier à la suite de chacun des coups possibles.

Pour modéliser le baguenaudier on utilise un tableau (de type list) de booléens. Une case vide est représentée par False et une case remplie par True.

L’exemple de situation du baguenaudier de 5 cases de la Figure 1 présentée plus haut sera ainsi représenté par le tableau suivant.

[False, False, True, True, False]

La fonction initialiser prend en paramètre un nombre entier n et elle renvoie une liste modélisant un baguenaudier vide de n cases.

Exemple :

>>> initialiser(3)
[False, False, False]

1. Écrire le code de la fonction initialiser

La fonction victoire renvoie un booléen indiquant si toutes les cases du baguenaudier sont remplies.

1 def victoire(tab):
2 for etat\_case in tab:
3 if etat\_case == ...:
4 return ...
5 ...

1. Recopier et compléter les lignes 3, 4 et 5 du code de la fonction victoire.

La fonction indice\_premiere\_case\_occupee prend en paramètre un tableau tab modélisant l’état du baguenaudier et renvoie l’indice de la première case remplie du baguenaudier, ou None si le baguenaudier est vide.

Exemples :

>>> indice\_premiere\_case\_occupee([True, True, True])
0
>>> indice\_premiere\_case\_occupee([False, False, False, True, True])
3
>>> indice\_premiere\_case\_occupee([False, False, False, False])
# pas de sortie car valeur None

1. Écrire le code de la fonction indice\_premiere\_case\_occupee.

La fonction coup\_valide prend en paramètres tab qui est une liste modélisant l’état d’un baguenaudier, et un entier case qui est l’indice de la case à jouer. La fonction renvoie True si le coup respecte les règles et False si ce n’est pas le cas.

L’indice de la première case du jeu est 0. Pour que le coup soit valide, il faut aussi s’assurer que l’indice est un entier positif inférieur à la longueur de tab.

1. Écrire le code la fonction coup\_valide.

La fonction changer\_case prend en paramètres un tableau tab modélisant l’état du baguenaudier et un indice de case nommé case. Si le coup désigné par case est valide, la fonction renvoie le tableau modélisant le baguenaudier obtenu après avoir changé l’état de la case correspondante. Si le coup joué n’est pas valide, le baguenaudier passé en paramètre est renvoyé sans modification.

Exemples :

>>> changer\_case([False, False, False], 1) # coup non valide
[False, False, False]
>>> changer\_case([False, False, False], 0) # coup valide
[True, False, False]
>>> changer\_case([False, True, False], 2) # coup valide
[False, True, True]

1. Écrire le code de la fonction changer\_case.

La résolution d’un baguenaudier de 3 cases, en Figure 2 et 4 cases, en Figure 3 sont données ci-dessous.

 

 Figure 2. Résolution du Figure 3. Résolution du

 baguenaudier à 3 cases baguenaudier à 4 cases

Il est possible d’obtenir la solution du jeu du baguenaudier, sous forme d’affichage, en utilisant des fonctions récursives nommées vider et remplir. Ces deux fonctions prennent en paramètre un entier n correspondant au nombre de cases du jeu. La fonction vider vide un baguenaudier de n cases initialement remplies. La fonction remplir remplit un baguenaudier de n cases initialement vides.

Le code de la fonction vider, incomplet, est donné ci-dessous. On notera que les cases sont numérotées à partir de 1 dans la suite et que l’appel vider(0)ne fait rien.

1 def vider(n):
2 if n == 1:
3 print('Vider case 1')
4 elif n > 1:
5 vider(n-2)
6 print(...)
7 remplir(n-2)
8 vider(n-1)

1. Recopier et compléter la ligne 6 du code de la fonction vider pour qu’elle affiche le texte 'Vider case ' suivi de la valeur de l’entier n.
2. En supposant que l’appel remplir(1) affiche ' Remplir case 1' et que l’appel remplir(0) ne fait rien, donner les affichages, dans la console, produits par l’exécution de vider(3).
3. En vous inspirant de la fonction vider, de la résolution donnée à la question précédente pour n = 3 et des Figures 2 et 3, écrire le code de la fonction récursive remplir.
4. On envisage d’utiliser la fonction vider pour un baguenaudier de 2000 cases. Expliquer si ce code est adapté à un baguenaudier de si grande taille. Justifier votre réponse.

# Exercice 3 (8 points)

*Cet exercice porte sur la programmation en Python, les bases de données relationnelles, le langage SQL, les systèmes d’exploitation et la sécurisation des communications.*

Pour simplifier la gestion de ses mots de passe, Alice décide de créer un gestionnaire de mots de passe regroupant les informations de connexion de chaque site qu’elle utilise.

## Partie A

Dans cette partie, on s’intéresse à la création des mots de passe d’Alice. On s’aidera de la table ASCII de la Figure 1 donnant le code ASCII en décimal et en hexadécimal des différents caractères utilisés dans ce sujet (lettres minuscules, lettres majuscules et les caractères spéciaux encadrés).



Figure 1. Table ASCII. Les caractères spéciaux utilisés dans ce sujet sont encadrés

Source : d’après https://fr.m.wikipedia.org/wiki/Fichier : ASCII-Table.svg

Alice souhaite créer une fonction Python gen\_mdp lui permettant de créer aléatoirement un mot de passe. Cette fonction :

* prend en paramètres :
	+ longueur (de type int) : le nombre de caractères du mot de passe ;
	+ cont\_min (de type bool) : un booléen indiquant si le mot de passe peut contenir des minuscules (True) ou non (False) ;
	+ cont\_maj (de type bool) : un booléen indiquant si le mot de passe peut contenir des majuscules (True) ou non (False) ;
	+ cont\_spe ( de type bool) : un booléen indiquant si le mot de passe peut contenir des caractères spéciaux (True) ou non (False) ;
* et renvoie un mot de passe (de type str) respectant les conditions définies par les paramètres.

On donne ci-dessous le code de la fonction gen\_mdp qui sera à compléter au fur et à mesure des questions.

 1 from random import randint
 2
 3 def gen\_mdp(longueur, cont\_min, cont\_maj, cont\_spe):
 4 # Pour qu'un mot de passe soit non vide, il doit
 5 # pouvoir contenir des minuscules ou des majuscules
 6 # ou des caractères spéciaux.
 7 assert (cont\_min or cont\_maj or cont\_spe)
 8 minuscules = [chr(i) for i in ...]
 9 majuscules = [...]
10 caracteres\_speciaux = ... + ...
11 jeu\_caracteres = []
12 if cont\_min:
13 ...
14 ...
15 ...
16 ...
17 ...
18 mot\_de\_passe = ''
19 n = len(jeu\_caracteres)
20 for i in range(longueur):
21 mot\_de\_passe = ...
22 return mot\_de\_passe

On rappelle que l’opérateur + permet en Python d’additionner deux éléments de type int, mais aussi de concaténer deux éléments de type list, ou encore deux éléments de type str.

On rappelle également que chr(i) renvoie la chaîne représentant un caractère dont le code ASCII est le nombre entier i.

Par exemple, chr(97) renvoie la chaîne de caractères 'a', tandis que chr(33) renvoie la chaîne de caractères '!'.

1. Alice s’inscrit sur un nouveau site lui demandant de créer un mot de passe de 8 caractères minimum, composé uniquement de majuscules et de minuscules. Écrire un appel à la fonction gen\_mdp qui permette de répondre aux exigences de ce site.

On admet que les caractères spéciaux sont uniquement ceux encadrés à la Figure 1.

1. Recopier et compléter les lignes 8 à 10 du code de la fonction gen\_mdp permettant d’initialiser les trois variables minuscules, majuscules et caracteres\_speciaux. On utilisera obligatoirement la syntaxe de tableaux donnés en compréhension pour cette initialisation.

Après cette initialisation, on doit avoir :

* + minuscules initialisée à ['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']
	+ majuscules initialisée à ['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']
	+ caracteres\_speciaux initialisée à ['!', '"', '#', '$', '%', '&', "'", '(', ')', '\*', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?', '@']
1. La variable jeu\_caracteres de la fonction gen\_mdp est une liste contenant tous les caractères autorisés pour fabriquer un nouveau mot de passe. Recopier et compléter la ligne 13 du code de la fonction gen\_mdp, puis écrire les autres lignes de code afin de créer la variable jeu\_caracteres. Le nombre de lignes de code à insérer est laissé à l’appréciation du candidat.
2. La documentation de la fonction randint indique que randint(a, b) renvoie un entier aléatoire compris entre les entiers a et b incluant les deux extrémités a et b. Recopier et compléter la ligne 21 du code de la fonction gen\_mdp.
3. Un site demande de créer un mot de passe possédant obligatoirement les caractéristiques suivantes :
	* le mot de passe doit contenir au moins 12 caractères ;
	* le mot de passe doit contenir au moins un caractère spécial et au moins une lettre minuscule.
* Expliquer en quoi le code de la fonction gen\_mdp peut renvoyer un mot de passe qui ne répond pas aux exigences du site.

## Partie B

Dans cette partie, on pourra utiliser les clauses du langage SQL pour :

* construire des requêtes d’interrogation à l’aide de SELECT, FROM, WHERE (avec les opérateurs logiques AND, OR), JOIN ... ON ;
* construire des requêtes d’insertion et de mise à jour à l’aide de UPDATE, INSERT, DELETE ;
* affiner les recherches à l’aide de DISTINCT, ORDER BY.

Alice souhaite mettre en œuvre une base de données composée des deux tables compte et site dont des extraits sont donnés dans les tableaux suivants.

| compte |
| --- |
| mot\_de\_passe | utilisateur | renouvellement | id\_site |
| Asrtg!Myfj | aliceB24 | 2022-06-30 | 1 |
| @rDfohpj!& | aliceB24 | 2021-03-12 | 2 |
| GxRGDxc(u-PM | alice\_B@votremailp.me | 2018-10-14 | 4 |
| Ghcj=+f\*AZs | alice1276 | 2022-06-30 | 3 |
| cYFgt!:Ehr; | alice\_B2@votremailp.me | 2022-06-30 | 4 |

| site |
| --- |
| id | nom\_site | url |
| 1 | Vosnotes | https://logi-educ.net/vosnotes/eleve.html |
| 2 | Banque Perso | https://www.banqueperso.fr/connexion.html |
| 3 | Elec verte | https://espace-client.ev.fr/login |
| 4 | Votremailp | https://account.votremailp.me/login |

* L’attribut mot\_de\_passe est une clé primaire de la table compte.
* L’attribut id est une clé primaire de la table site.
* L’attribut renouvellement, correspondant au dernier renouvellement du mot de passe, est une chaîne de caractères de format AAAA-MM-JJ. Ainsi un mot de passe renouvelé le 21 février 2025 correspond à un attribut renouvellement de '2025-02-21'.
* Dans la table compte, l’attribut id\_site est une clé étrangère référençant l’attribut id de la table site.
1. Expliquer en quoi il n’est pas possible, pour Alice, d’avoir le même mot de passe pour deux sites différents.
2. Écrire la requête SQL permettant d’afficher toutes les URL enregistrées dans la base de données.
3. La Banque Perso a demandé à Alice de renouveler son mot de passe. Elle remplace le mot de passe '@rDfohpj!&' par 'yhTS?d@UTJe'. Écrire la requête SQL permettant de faire la modification dans la base de données. Dans cette question, on ignorera l’attribut renouvellement et on ne cherchera pas à le mettre à jour.

On rappelle que le langage SQL utilise l’ordre lexicographique (ordre du dictionnaire) pour comparer deux éléments de type texte. Selon cet ordre, on a par exemple '3' supérieur à '1' et 'python' est inférieur à 'sql'.

1. À la date du 20 mars 2025, Alice a décidé de lister tous les mots de passe qui n’ont pas été renouvelés depuis plus d’un an. Écrire la requête SQL permettant de donner la liste des id\_site concernés.
2. Donner une raison pour laquelle Alice a préféré choisir le format AAAA-MM-JJ pour l’attribut renouvellement plutôt que le format JJ-MM-AAAA.
3. Écrire une requête SQL permettant d’afficher tous les utilisateurs et les mots de passe du ou des sites de nom 'Votremailp' dont l’identifiant est supposé non connu. Le résultat devra être affiché par ordre chronologique de date de renouvellement de mot de passe.
4. Pour réaliser le projet, il aurait été possible de regrouper toutes les informations dans une seule table mais Alice a choisi d’utiliser deux tables : les tables site et compte. Donner un avantage impliqué par le choix d’Alice.

## Partie C

Dans cette partie, on s’intéresse à la sécurité du gestionnaire de mots de passe d’Alice. Alice utilise un système d’exploitation multi-utilisateurs basé sur Linux et elle a les droits de lecture, d’écriture et d’exécution dans son répertoire personnel /home.

La base de données est stockée dans un fichier gestionnaire.db.
Alice décide de chiffrer ce fichier. Elle crée un fichier chiffrement.py dans lequel elle écrit le code d’une fonction Python chiffrement dont la documentation est donnée ci-après.

1 def chiffrement(f\_source, f\_dest, f\_cle):
2 '''Crée un fichier f\_dest contenant les données
3 du fichier f\_source chiffrées selon la clé
4 contenue dans le fichier f\_cle.
5 f\_source (str) : chemin vers le fichier à chiffrer
6 f\_dest (str) : chemin vers le fichier chiffré
7 f\_cle (str) : chemin vers le fichier contenant la clé
8 '''

On donne l’arborescence de fichiers utilisée par Alice sur la Figure 2.



Figure 2. Arborescence de fichiers

Alice ouvre un interpréteur Python dans le répertoire Documents, qui est ainsi le répertoire courant (ou répertoire de travail) et exécute le contenu du fichier chiffrement.py.

1. Écrire l’appel à la fonction chiffrement permettant, à l’aide de la clé contenue dans le fichier cle, de chiffrer le fichier gestionnaire.db en créant le fichier chiffré secret.db dans le répertoire Perso.

L’algorithme de chiffrement consiste à appliquer l’opération OU-exclusif bit à bit entre le fichier source et le fichier contenant la clé de chiffrement.

Le premier bit du fichier chiffré est le résultat du OU-exclusif entre le premier bit du fichier source et le premier bit du fichier clé. De même, le deuxième bit est obtenu par le OU-exclusif entre les deuxièmes bits de ces deux fichiers et ainsi de suite.

On suppose, pour simplifier, que le fichier contenant la clé est de la même taille que le fichier source, qui est donc aussi la taille du fichier chiffré.

On rappelle ci-après la table de vérité du OU-exclusif, noté XOR.

| Table de vérité du XOR |
| --- |
| a | b | a XOR b |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

On rappelle que l’hexadécimal correspond à l’écriture des entiers dans la base 16. Les chiffres de cette base sont notés 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

1. Donner le résultat, en écriture hexadécimale, de l’opération OU-exclusif bit à bit entre deux octets dont les écritures hexadécimales sont respectivement A3 et 59.
2. Montrer que, pour tous bits a et b, on a la propriété suivante :
(a XOR b) XOR b = a.
3. Indiquer, en justifiant, si le chiffrement utilisé par Alice est symétrique ou asymétrique.

Alice exécute la commande ls -l secret.db et obtient la réponse donnée ci-dessous :

-rw-r--r-- 1 alice eleves 42480 mars 25 11:57 secret.db

1. Justifier le fait qu’un attaquant a le champ libre pour tenter un déchiffrement du fichier secret.db et expliquer ce que devrait faire Alice pour corriger ce problème.

## Partie D

On donne ci-dessous quatre bonnes pratiques proposées par le gouvernement en matière de gestion des mots de passe.

* P1 : Utilisez un mot de passe différent pour chaque accès (messagerie, banque en ligne, comptes de réseaux sociaux, etc.) : en cas de compromission de l’un de vos comptes, cela évitera l’effet boule de neige.
* P2 : Créez un mot de passe suffisamment long, complexe et inattendu, de 8 caractères minimum, contenant des minuscules, des majuscules, des chiffres et des caractères spéciaux.
* P3 : Ne communiquez jamais votre mot de passe à un tiers : aucune organisation ou personne de confiance ne vous demandera de lui communiquer votre mot de passe.
* P4 : Utilisez un gestionnaire de mots de passe : pas simple de retenir tous ses codes de connexion ! Heureusement des outils de type « coffres forts de mots de passe » existent. Ces derniers mémorisent tous vos mots de passe et vous permettent d’en générer de manière aléatoire.

Source : d’après https://cyber.gouv.fr/bonnes-pratiques-protegez-vous

1. En justifiant votre réponse, préciser en quoi les choix d’Alice dans la conception de son gestionnaire de mots de passe respectent ou non chacune des quatre bonnes pratiques mentionnées.