ACADÉMIE
DE REIMS
Liberté
Égalité
Fraternité

NUMERIQUE
et
SCIENCES
INFORMATIQUES

Épreuve de l'enseignement de spécialité

Sujet d'entraînement

Partie Pratique

Classe Terminale de la voie générale

Le candidat doit traiter les 2 exercices

Ce sujet comporte 4 pages

Exercice 1 (4 points)
Vous avez à gérer le réseau local d'une entreprise s'appelant *forVIP*.
Pour cela, vous devez construire une fonction `octet_vers_dec` :

- qui prend en paramètre une liste `lst_bits` formée de huit entiers, chaque entier valant 0 ou 1 (cette liste implémente un octet),
- qui renvoie le nombre entier, compris entre 0 et 255, qui correspond à l'octet de la liste saisie.

Vérifier entre autre que :

```
>>> octet_vers_dec([0,0,0,0,1,1,0,1])
    13
```

Une fois la fonction réalisée, pensez à rajouter deux préconditions à la fonction pour vérifier l'argument saisie par l'utilisateur.

Exercice 2 (4 points)
Vous avez à gérer le réseau local d'une entreprise s'appelant *forVIP*.
Pour cela, vous définissez une classe `Adresse4VIP` gérant les adresses IPv4 de ce réseau.
On rappelle qu'une adresse IPv4 est une adresse de longueur 4 octets, chaque octet étant écrit en valeur décimale et étant séparé des autres par un point.
Ce réseau privé d'entreprise est constitué des adresses IP allant de 172.25.3.0 à 172.25.3.255.
Les adresses IP 172.25.3.0 et 172.25.3.255 sont des adresses réservées pour le réseau lui-même et le broadcast. Les autres peuvent être attribuées à des postes informatiques.
L'attribut `attribuee` est un booléen valant `True` si l'adresse considérée est attribuée à un poste informatique de l'entreprise et `False` sinon.

Compléter le code ci-dessous et instancier trois objets : `adresse1`, `adresse2`, `adresse3` avec respectivement les arguments suivants : `'172.25.3.1'`, `'172.25.3.4'` et `'172.25.3.255'`.

```
class Adresse4VIP:
    def __init__(self, adresse:str, booleen=True):
        """ constructeur"""
        self.adresse = ...    # l'adresse saisie est de type chaîne de caractères
        self.attribuee = booleen  # True lorsque l'adresse est attribuée à un poste informatique

    def liste_octet(self):
        """renvoie une liste de nombres entiers,
           la liste des octets de l'adresse IP"""
        return [int(i) for i in self.adresse.split(".")]

    def est_reservee(self):
        """renvoie True si l'adresse IP est une des deux adresses
           réservées, False sinon"""
        return ...

    def get_attribuee(self):
        """ renvoie la valeur de l'attribut attribuee"""
        ...

    def liberer(self):
        """ Dans les cas des adresses non réservées :
        modifie l'attribut attribuee pour signifier que l'adresse IP n'est plus attribuée"""
        if ...:
            self.attribuee = False

    def obtenir_adresse_suivante(self):
         """renvoie
           * l'adresse self si celle-ci n'est pas encore attribuée
           * l'adresse IP qui suit l'adresse self
           si l'adresse suivante est possible et si l'adresse self est déjà attribuée
           * 'impossible' sinon"""
        if not self.attribuee:
            return ...
        elif ...<254 :
            octet_nouveau = ... + ...
            return '172.25.3.' + ...
        else:
            return 'impossible'
```

Vérifier entre autre que :

```
>>> adresse2.get_attribuee()
    True
>>> adresse2.liberer()
>>> adresse2.get_attribuee()
    False
>>> adresse3.est_reservee()
    True
>>> adresse1.obtenir_adresse_suivante()
    '172.25.3.2'
>>> adresse2.obtenir_adresse_suivante()
    '172.25.3.4'
```