BACCALAURÉAT

SESSION 2024

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°39

DURÉE DE L'ÉPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (10 points)

Écrire une fonction `recherche` qui prend en paramètres `elt` un nombre entier et `tab` un tableau de nombres entiers (type `list`), et qui renvoie l'indice de la dernière occurrence de `elt` dans `tab` si `elt` est dans `tab` et `None` sinon.

Exemples :

```
>>> recherche(1, [2, 3, 4]) # renvoie None
>>> recherche(1, [10, 12, 1, 56])
2
>>> recherche(1, [1, 0, 42, 7])
0
>>> recherche(1, [1, 50, 1])
2
>>> recherche(1, [8, 1, 10, 1, 7, 1, 8])
5
```

EXERCICE 2 (10 points)

On définit une classe gérant une adresse IPv4.

On rappelle qu'une adresse IPv4 est une adresse de longueur 4 octets, notée en décimale à point, en séparant chacun des octets par un point. On considère un réseau privé avec une plage d'adresses IP de `192.168.0.0` à `192.168.0.255`.

On considère que les adresses IP saisies sont valides.

Les adresses IP `192.168.0.0` et `192.168.0.255` sont des adresses réservées.

Le code ci-dessous implémente la classe `AdresseIP`.

```
class AdresseIP:
    def __init__(self, adresse):
        self.adresse =...

    def liste_octets(self):
        """renvoie une liste de nombres entiers,
        la liste des octets de l'adresse IP"""
        # Note : split découpe la chaine de caractères
        # en fonction du séparateur
        return [int(i) for i in self.adresse.split(".")]

    def est_reservee(self):
        """renvoie True si l'adresse IP est une adresse
        réservée, False sinon"""
        reservees = [ ... ]
        return ...

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
        IP qui suit l'adresse self si elle existe et None sinon"""
        octets = ...
        if ... == 254:
            return None
        octet_nouveau = ... + ...
        return AdresseIP('192.168.0.' + ...)
```

Compléter le code ci-dessus et instancier trois objets : `adresse1`, `adresse2`, `adresse3` avec respectivement les arguments suivants :

`'192.168.0.1','192.168.0.2','192.168.0.0'`

Vérifier que :

```
>>> adresse1.liste_octets()
[192, 168, 0, 1]
>>> adresse1.est_reservee()
False
>>> adresse3.est_reservee()
True
>>> adresse2.adresse_suivante().adresse # acces valide à adresse
                # ici car on sait que l'adresse suivante existe
'192.168.0.3'
```