BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°32

DUREE DE L'EPREUVE : 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 (4 points)

Écrire une fonction `recherche` qui prend en paramètres `elt` un nombre et `tab` un tableau de nombres, et qui renvoie l'indice de la dernière occurrence de `elt` dans `tab` si `elt` est dans `tab` et le `-1` sinon.

Exemples :

```
>>> recherche(1,[2,3,4])
-1
>>> recherche(1,[10,12,1,56])
2
>>> recherche(1,[1,50,1])
2
>>> recherche(1,[8,1,10,1,7,1,8])
5
```

EXERCICE 2 (4 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_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 adresse
           réservée, False sinon"""
        return ... or ...

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
           IP qui suit l'adresse self
           si elle existe et False sinon"""
        if ... < 254:
            octet_nouveau = ... + ...
            return AdresseIP('192.168.0.' + ...)
        else:
            return False
```

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.est_reservee()
False
>>> adresse3.est_reservee()
True
>>> adresse2.adresse_suivante().adresse
'192.168.0.3'
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 32 des épreuves pratiques NSI 2022

Remarques:
    * /!\ attention ici c'est la DERNIÈRE occurrence qui est recherchée !
"""


def recherche(elt, tab):
    """ Recherche la valeur de la dernière occurrence
    de elt dans le tableau.

    Args:
        elt (int) : nombre à rechercher

    Returns:
        int : indice de la dernière valeur recherchée ou
              -1 si absent du tableau

    Tests et Exemples:
    >>> recherche(1, [2, 3, 4])
    -1
    >>> recherche(1, [10, 12, 1, 56])
    2
    >>> recherche(1, [1, 50, 1])
    2
    >>> recherche(1, [8, 1, 10, 1, 7, 1, 8])
    5
    """
    n = len(tab)

    # BOUCLE
    # -> invariant: i_elt est l'indice de la dernière apparition de elt
    # dans la zone tab[0 .. i-1] ou -1 si elt n'y est pas
    i_elt = -1

    # -> condition d'arrêt: après la boucle i <- n
    for i in range(n):
        if tab[i] == elt:
            i_elt = i

    # fin BOUCLE
    # tout le tableau est parcouru
    return i_elt


# vérification avec des assertions
assert recherche(1, [2, 3, 4]) == -1
assert recherche(1, [10, 12, 1, 56]) == 2
assert recherche(1, [1, 50, 1]) == 2
assert recherche(1, [8, 1, 10, 1, 7, 1, 8]) == 5

# vérification avec des affichages
print(recherche(1, [2, 3, 4]))
print(recherche(1, [10, 12, 1, 56]))
print(recherche(1, [1, 50, 1]))
print(recherche(1, [8, 1, 10, 1, 7, 1, 8]))

# vérification avec doctest
from doctest import testmod
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 32 des épreuves pratiques NSI 2022

Remarques:
    * /!\ ne pas oublier de convertir en str pour concaténer
    pour concaténer des chaînes de caractère on utilise l'opérateur `+`
    mais il FAUT que chaque argument soit une chaine.
"""


class AdresseIP:

    def __init__(self, adresse):
        self.adresse = adresse

    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 adresse
           réservée, False sinon"""
        return self.adresse == '192.168.0.0' or self.adresse == '192.168.0.255'

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
           IP qui suit l¿adresse self
           si elle existe et False sinon"""

        # la liste_octet contient dans sa 4ème case le dernier octet
        if self.liste_octet()[3] < 254:
            # incrémenter le dernier octet
            octet_nouveau = self.liste_octet()[3] + 1
            # concaténer les chaines de caractère
            return AdresseIP('192.168.0.' + str(octet_nouveau))
        else:
            return False



# Vérification avec des affichages
adresse1 = AdresseIP('192.168.0.1')
adresse2 = AdresseIP('192.168.0.2')
adresse3 = AdresseIP('192.168.0.0')
print(adresse1.est_reservee())
print(adresse3.est_reservee())
print(adresse2.adresse_suivante().adresse)

# Vérification avec des assertions
adresse1 = AdresseIP('192.168.0.1')
adresse2 = AdresseIP('192.168.0.2')
adresse3 = AdresseIP('192.168.0.0')
assert adresse1.est_reservee() == False
assert adresse3.est_reservee() == True
assert adresse2.adresse_suivante().adresse == '192.168.0.3'
```