Correction NSI 32

EXERCICE 1 :

```
1  def recherche(elt,tab):
2      if elt in tab:
3          for k in range(len(tab)):
4              if elt == tab[k]:
5                  ind_der_occu = k
6      else :
7          ind_der_occu = -1
8      return ind_der_occu
```

-On crée la fonction **recherche** avec comme paramètre **elt(element)** et **tab(tableau).** (demandé dans le sujet)

-On utilise **if** pour savoir si **elt (l'élément)** est présent dans **tab (le tableau).**
Si il est présent, on fait une **boucle** qui va s'exécuter **autant de fois qu'il y a d'éléments** dans **le tableau (tab)** où on va utiliser un **if** qui va chercher si **elt (l'élément qu'on cherche)** est présent dans le tableau **au rang k** qui va de 0 au nombre d'éléments dans **tab**. Donc si **elt** est dans le tableau **au rang k**, la variable **ind_der_occu** ,qui signifie l'indice de la dernière occurrence, va être égale à **k**.

-On utilise **else** pour que si **elt** n'est pas dans **tab** alors la variable **ind_der_occu** soit égale à -1. (demandé dans le sujet)

-Puis pour finir on renvoie la variable **ind_der_occu** pour avoir l'indice où l'élément recherché dans le tableau apparaît pour la dernière fois (dernière occurrence).

EXERCICE 2 :

```
class AdresseIP:

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

-On met self.adresse = adresse car c'est ainsi qu'il est nommé dans les paramètres de la méthode __init__().

```
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'
```

-Nous sommes face à un “or”, de ce fait il faut mettre la condition des deux côtés.
Ici, les adresse réservée étaient données dans le sujet, il nous suffit de les ajouter sous forme de chaîne de caractères(str).

```
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 self.liste_octet()[-1] < 254:
        octet_nouveau = self.liste_octet()[-1] + 1
        return AdresseIP('192.168.0.' + str(octet_nouveau))
    else:
        return False
```

-Ici, “liste_octet” est une méthode, de ce fait nous devons l'appeler obligatoirement avec des parenthèses “()”. Cependant, étant une liste, nous pouvons ensuite ajouter des crochets ”[n]” avec l'indice ‘n’ recherché, ici c'est le dernier donc -1.

-Ensuite pour la variable octet_nouveau, on ajoute dans le remplacement de octet_nouveau la condition précédente c'est-à-dire self.liste_octet()[-1] car c'est ce que l'on veut modifier (ici le dernier octet de l'adresse IP).

-Pour le return, octet_nouveau, réalisé juste avant est un résultat int et non str, de ce fait on doit transformer la variable octet_nouveau en chaîne de caractères (str).

```
adresse1=AdresseIP('192.168.0.1')
adresse2=AdresseIP('192.168.0.2')
adresse3=AdresseIP('192.168.0.0')
```

-Attention, nous sommes sortis de la classe AdresseIP, de ce fait nous devons l'appeler pour que les critères de l'adresse IP respectent les paramètres de la classe et donc tout ce que l'on a fait avant. De ce fait, nous devons mettre AdresseIP(‘adresse’) pour qu'elle soit bien prise en compte sans bug.