Corrigé Sujet 8 épreuve pratique NSI

Michel Billaud (michel.billaud@laposte.net)

1er février 2022

Table des matières

1 Licence

- Les notes sont publiées dans https://www.mbillaud.fr/notes/
- Sources dans https://github.com/MichelBillaud/notes-diverses

2 Le sujet

Se trouve sur la page https://eduscol.education.fr/2661/banque-des-epreuves-pratiques-de-specialite-nsi, dans https://eduscol.education.fr/document/33202/download

- recherche d'une occurrence dans une liste
- insertion dans un tableau ordonnée

3 Recherche d'une occurrence dans une liste

Sujet : retourner l'indice de la première occurrence d'un élément dans un tableau, -1 si il est absent

3.1 Solution : boucle sur les indices

Il est naturel de penser à faire une boucle sur les indices valides. Chercher l'indice de la **première** valeur suggère de sortir de cette boucle, et pourquoi ne pas en profiter pour retourner l'indice ?

```
def recherche(element, tableau):
    for i in range(len(tableau)):
        if tableau[i] == element:
            return i
    return -1

print ("Tests")
assert recherche(1, [2, 3, 4])       == -1
assert recherche(1, [10, 12, 1, 56]) == 2
assert recherche(50, [1, 50, 1])     == 1
assert recherche(15, [8, 9, 10, 15]) == 3
print ("OK")
```

3.2 Solution : boucle sur les valeurs

En prenant le problème différemment, on peut aussi faire une boucle sur les valeurs (`for v in tableau`), en contrôlant manuellement un indice

```
def recherche(element, tableau):
    indice = 0
    for v in tableau:
        if v == element:
            return indice
        indice += 1
    return -1
```

3.3 Solution : en se compliquant la vie (boucle `while`)

```
def recherche(element, tableau):
    taille = len(tableau)
    indice = 0
    while indice < taille and tableau[indice] != element:
        indice += 1
    if indice == taille:
        return -1
    else:
        return indice
```

3.4 Solution : encore pire (boucle + booléen)

```
def recherche(element, tableau):
    taille = len(tableau)
    indice = 0
    trouve = False
    while indice < taille and not trouve:
        if tableau[indice] == element:
            trouve = True
        else:
            indice += 1
    if trouve:
        return indice
    else:
        return -1
```

L'affirmation péremptoire "c'est encore pire" est justifiée par la comparaison sur

- le nombre de variables introduites ;
- le nombre d'instructions ;

pour arriver au même résultat.

4 Exercice 2 : insertion dans une liste ordonnée

Code fourni

```
def insere(a, tab):
    l = list(tab) #l contient les mêmes éléments que tab
    l.append(a)
    i = ...                         # 1
    while a < ... and i >= 0:       # 2
        l[i+1] = ...                # 3
        l[i] = a
        i = ...                     # 4
    return l
```

4.1 Résolution

Les deux premières instructions ajoutent l'élément, au bout d'une copie de la liste fournie en paramètre.

- Il se peut que cet élément soit à sa place, si il est plus grand que son prédécesseur. Ou pas.

- Si il ne l'est pas, on décale son prédécesseur. Il a peut être trouvé sa place définitive. Ou pas.
- Sinon on recommence, etc.

Ca s'arrêtera de toutes facons en arrivant tout a fait à gauche, quand il n'y a plus de prédécesseur.

Prêtons attention aux indices

- le corps de la boucle sert à décaler le prédécesseur.
- ce prédécesseur porte l'indice `i` (on affecte `a` à cet endroit)

Les deux premières instructions du corps de boucle sont donc

```
l[i+1] = l[i]        # 3 décalage
l[i]   = a           #   remplissage
```

- Pour avoir une chance de décaler dans le dernier élément `l[len(l) - 1]`, il faut que `i+1` puisse prendre la valeur `len(l)-1`. La première valeur pour `i` devra donc être `len(l) - 2 # 1`

et bien sûr à chaque tour on va vers la gauche, en diminuant `i` : `i = i - 1 #` 4.

Reste la condition : si on déplace `l[i]`, c'est parce qu'on a détecté qu'il n'était pas à sa place par rapport à `a`, après comparaison `a < l[i]`

4.2 Code complété

```
def insere(a, tab):
    l = list(tab) #l contient les mêmes éléments que tab
    l.append(a)
    i = len(l) - 2                # 1
    while a < l[i] and i >= 0:    # 2
        l[i+1] = l[i]             # 3
        l[i]   = a
        i = i - 1                 # 4
    return l

print ("Tests insertion")
assert insere(3,[1,2,4,5]) == [1, 2, 3, 4, 5]
assert insere(10,[1,2,7,12,14,25]), [1, 2, 7, 10, 12, 14, 25]
assert insere(1,[2,3,4]) == [1, 2, 3, 4]
print ("OK");
```