2021 - J2 - Métropole 2

Exercice 4

1 - Le code proposé n'effectue pas l'échange car dès la première ligne de la fonction, les cases `i2` et `i1` du tableau contiennent toutes les deux la même valeur : le contenu initial de la case `i1`.

L'exécution de la deuxième ne change absolument rien à l'état de la mémoire. Ce n'est donc pas un échange.

2 - Un appel à la commande `randint(0, 10)` renvoie tout nombre entier compris entre 0 (inclus) et 10 (inclus aussi !).

Donc on peut obtenir : `0`, `1`, `9` et `10`.

3.a - La fonction `melange` est récursive. Pour qu'elle se termine il faut montrer que le cas de base est atteignable dans tous les cas.

Ici le cas de base arrive lorsque l'argument `ind` vaut `0` ou moins. Or `ind` est un nombre entier et chaque appel récursif se fait avec une valeur de `ind` diminuée de 1. Donc nécessairement, après `ind - 1` appels récursif, la fonction `melange` effectue l'appel `melange(lst, 0)`. Cet appel sera le dernier car le cas de base sera atteint.

3.b - Le cas de base est atteint lorsque `ind` vaut `0`. Pour une liste de longueur n, l'appel initial est `melange(lst, n - 1)`. Comme à chaque appel `ind` diminue de 1, il faut donc $n - 1$ appels pour arriver au cas de base.

3.c - Initialement, on a `lst` $\leftarrow$ `[0, 1, 2, 3, 4]`.

Détaillons la première exécution. L'appel `melange(lst, 4)` effectue un premier tirage aléatoire et on obtient `j`$\leftarrow$`2`. Ainsi il y a un échange entre les valeurs des indices `4` et `2` de `lst`. Désormais on a `lst` $\leftarrow$ `[0, 1, 4, 3, 2]` et l'appel récursif est `melange(lst, 2)`.

Cette première exécution correspond à la première ligne du tableau. Les quatres appels récursifs qui suivent sont sur les quatre lignes suivantes.

Les affichages demandés sont ceux de la première colonne.

lst	ind	j	appel récursif
[0, 1, 2, 3, 4]	4	2	melange([0, 1, 4, 3, 2], 3)
[0, 1, 4, 3, 2]	3	1	melange([0, 3, 4, 1, 2], 2)
[0, 3, 4, 1, 2]	2	2	melange([0, 3, 4, 1, 2], 1)
[0, 3, 4, 1, 2]	1	0	melange([3, 0, 4, 1, 2], 0)
[3, 0, 4, 1, 2]			

3.d - Voici une version itérative du mélange de Fischer Yates :

[62]:
```
def melange(lst):
    print(lst)

    # dernier indice du tableau
    n = len(lst)

    # parcours du tableau en commençant par la fin
    # et en s'arrêtant à l'indice 0 exclu
    for ind in range(n-1, 0, -1):
        j = randint(0, ind)
        echange(lst, ind, j)
        print(lst)
```

```
[0, 1, 2, 3, 4]
[0, 1, 4, 3, 2]
[0, 3, 4, 1, 2]
[0, 4, 3, 1, 2]
[0, 4, 3, 1, 2]
```