Corrigé

1. Le code proposé ne permet pas l'échange des éléments d'indices i_1 et i_2 car la première ligne de la fonction `echange` écrase la valeur d'indice i_2 et la remplace par l'élément d'indice i_1. À l'issue de la deuxième ligne, les deux éléments d'indice i_1 et i_2 sont donc égaux : ils ne sont pas échangés.

   Pour échanger les deux éléments, deux solutions : ou bien on passe par une variable intermédiaire pour conserver en mémoire la valeur initiale de l'élément d'indice i_2, ou bien on procède à une double affectation.

   ```
   # Version avec variable intermédiaire

   def echange(lst, i1, i2):
       rec = lst[i2]
       lst[i2] = lst[i1]
       lst[i1] = rec

   # Version avec double affectation

   def echange(lst, i1, i2):
       lst[i1], lst[i2] = lst[i2], lst[i1]
   ```

   Test de ces deux fonctions :

   ```
   >>> liste = [1,2,3]
   >>> echange(liste,0,2)
   >>> liste
   [3, 2, 1]
   ```

2. D'après la documentation `randint(0, 10)` renvoie un entier compris entre 0 inclus et 10 inclus. Les valeurs possibles sont donc 0, 1, 9 et 10.

3. **a.** La fonction `melange` est une fonction **récursive** car elle s'appelle elle-même (dernière ligne). Dans l'hypothèse où, comme dit dans la question suivante, la variable `ind` est un entier initialement égal au plus grand indice possible de la liste `lst`, chaque appel récursif est effectué en diminuant cet indice d'une unité. On arrivera donc nécessairement, après un nombre **fini** d'appel à une valeur de `ind` égale à 0. Dans ce cas, la condition `ind > 0` n'étant plus vérifiée, la fonction `melange` se terminera sans rien faire : c'est le cas de base.

   b. La fonction est exécutée avec `ind = n-1`. Le premier appel récursif est fait avec `ind = n - 2`, le second avec `ind = n - 3`, ... et le dernier avec `ind = 0`. Il y a donc $n-1$ appels récursifs, sans compter l'appel initial.

   c. Nous avons initialement `lst = [0, 1, 2, 3, 4]` comme indiqué dans l'énoncé. La première valeur aléatoire renvoyée est 2. Le dernier élément de la liste est donc échangé avec l'élément d'indice 2. On obtient donc `lst = [0, 1, 4, 3, 2]` : c'est bien ce qu'indique l'énoncé.

   La deuxième valeur aléatoire est égale à 1 et la fonction est appelée avec `ind = 3`. On échange donc l'élément d'indice trois avec celui d'indice un. D'où l'affichage `[0, 3, 4, 1, 2]`.

   La valeur suivante renvoyée par `randint` est 2 et la fonction est appelée avec `ind = 2`. On échange donc l'élément d'indice deux avec celui d'indice deux. Rien ne change. D'où l'affichage `[0, 3, 4, 1, 2]`.

   La valeur suivante renvoyée par `randint` est 0 et la fonction est appelée avec `ind = 1`. On échange donc l'élément d'indice zéro avec celui d'indice un. D'où l'affichage `[3, 0, 4, 1, 2]` et le programme se termine.

   En résumé, l'affichage obtenu sera le suivant :

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

d. Version itérative de l'algorithme utilisant une boucle `for` :

```
def melange(lst):
    for ind in range(len(lst) - 1, -1, -1):
        print(lst)
        j = randint(0, ind)
        echange(lst, ind, j)
```

Version avec une boucle `while` :

```
def melange(lst):
    ind = len(lst) - 1
    while ind > 0:
        print(lst)
        j = randint(0, ind)
        echange(lst, ind, j)
        ind = ind - 1
```