

# Exercice type bac : récursivité

## EXERCICE 1 (4 points)

Cet exercice porte sur la programmation en général et la récursivité en particulier.

On s'intéresse dans cet exercice à un algorithme de mélange des éléments d'une liste.

1. Pour la suite, il sera utile de disposer d'une fonction `echange` qui permet d'échanger dans une liste `lst` les éléments d'indice `i1` et `i2`. Expliquer pourquoi le code Python ci-dessous ne réalise pas cet échange et en proposer une modification. []
2. La documentation du module `random` de Python fournit les informations ci-dessous concernant la fonction `randint(a,b)` :

```
pour randrange(a,b+1).
```

Parmi les valeurs ci-dessous, quelles sont celles qui peuvent être renvoyées par l'appel `randint(0, 10)` ?

0	1	3.5	9	10	11
---	---	-----	---	----	----

3. Le mélange de Fischer Yates est un algorithme permettant de permuter aléatoirement les éléments d'une liste. On donne ci-dessous une mise en œuvre récursive de cet algorithme en Python.

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

- a. Expliquer pourquoi la fonction `melange` se termine toujours.
- b. Lors de l'appel de la fonction `melange`, la valeur du paramètre `ind` doit être égal au plus grand indice possible de la liste `lst`. Pour une liste de longueur `n`, quel est le nombre d'appels récursifs de la fonction `melange` effectués, sans compter l'appel initial ?
- c. On considère le script ci-dessous :

```
lst = [v for v in range(5)]  
melange(lst, 4)
```

On suppose que les valeurs successivement renvoyées par la fonction `randint` sont 2, 1, 2 et 0.

Les deux premiers affichages produits par l'instruction `print(lst)` de la fonction `melange` sont :

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

Donner les affichages suivants produits par la fonction `melange`.

d. Proposer une version itérative du mélange de Fischer Yates.

## EXERCICE 2 (4 points)

Cet exercice traite du thème «programmation», et principalement de la récursivité.

On rappelle qu'une chaîne de caractères peut être représentée en Python par un texte entre guillemets "" et que :

- la fonction `len` renvoie la longueur de la chaîne de caractères passée en paramètre ;
- si une variable `ch` désigne une chaîne de caractères, alors `ch[0]` renvoie son premier caractère, `ch[1]` le deuxième, etc. ;
- l'opérateur `+` permet de concaténer deux chaînes de caractères.

**Exemples :**

```
>>> texte = "bricot"  
>>> len(texte)  
6  
>>> texte[0]  
"b"  
>>> texte[1]  
"r"  
>>> "a" + texte  
"abricot"
```

On s'intéresse dans cet exercice à la construction de chaînes de caractères suivant certaines règles de construction.

**Règle A:** une chaîne est construite suivant la règle A dans les deux cas suivants:

- soit elle est égale à "a" ;
- soit elle est de la forme "a" + chaîne + "a", où chaîne est une chaîne de caractères construite suivant la règle A.

**Règle B :** une chaîne est construite suivant la règle B dans les deux cas suivants :

- soit elle est de la forme "b"+chaîne+"b", où chaîne est une chaîne de caractères construite suivant la règle A ;
- soit elle est de la forme "b" + chaîne+"b", où chaîne est une chaîne de caractères construite suivant la règle B.

On a reproduit ci-dessous l'aide de la fonction choice du module random.

```
>>>from random import choice
>>>help(choice)
Help on method choice in module random:
choice(seq) method of random.Random instance
Choose a random element from a non-empty sequence.
```

La fonction A() ci-dessous renvoie une chaîne de caractères construite suivant la règle A, en choisissant aléatoirement entre les deux cas de figure de cette règle.

```
def A():
    if choice([True, False]):
        return "a"
    else:
        return "a" + A() + "a"
```

1. a. Cette fonction est-elle récursive ? Justifier.

b. La fonction choice([True, False]) peut renvoyer False un très grand nombre de fois consécutives. Expliquer pourquoi ce cas de figure amènerait à une erreur d'exécution.

Dans la suite, on considère une deuxième version de la fonction A. À présent, la fonction prend en paramètre un entier n tel que, si la valeur de n est négative ou nulle, la fonction renvoie "a". Si la valeur de n est strictement positive, elle renvoie une chaîne de caractères

construite suivant la règle A avec un n décrémenté de 1, en choisissant aléatoirement entre les deux cas de figure de cette règle.

```
def A(n):  
    if ... or choice([True, False]) :  
        return "a"  
    else:  
        return "a" + ... + "a"
```

2. a. Recopier sur la copie et compléter aux emplacements des points de suspension ... le code de cette nouvelle fonction A.

b. Justifier le fait qu'un appel de la forme A(n) avec n un nombre entier positif inférieur à 50, termine toujours.

On donne ci-après le code de la fonction récursive B qui prend en paramètre un entier n et qui renvoie une chaîne de caractères construite suivant la règle B.

```
def B(n):  
    if n <= 0 or choice([True, False]):  
        return "b" + A(n-1) + "b"  
    else:  
        return "b" + B(n-1) + "b"
```

On admet que : • les appels A(-1) et A(0) renvoient la chaîne "a"; • l'appel A(1) renvoie la chaîne "a" ou la chaîne "aaa"; • l'appel A(2) renvoie la chaîne "a", la chaîne "aaa" ou la chaîne "aaaaa".

3. Donner toutes les chaînes possibles renvoyées par les appels B(0), B(1) et B(2).

On suppose maintenant qu'on dispose d'une fonction raccourcir qui prend comme paramètre une chaîne de caractères de longueur supérieure ou égale à 2, et renvoie la chaîne de caractères obtenue à partir de la chaîne initiale en lui ôtant le premier et le dernier caractère.

Exemple :

```
>>> raccourcir("abricot")  
"brico"  
>>> raccourcir("ab")  
""
```

4. a. Recopier sur la copie et compléter les points de suspension ... du code de la fonction regleA ci-dessous pour qu'elle renvoie True si la chaîne passée en paramètre est

construite suivant la règle A, et `False` sinon.

```
def regleA(chaine):  
    n = len(chaine)  
    if n >= 2:  
        return chaine[0] == "a" and chaine[n-1] == "a" and  
            regleA(...)   
    else:  
        return chaine == ...
```

b. Écrire le code d'une fonction `regleB`, prenant en paramètre une chaîne de caractères et renvoyant `True` si la chaîne est construite suivant la règle B, et `False` sinon.