



Exercice 1 On donne le programme :

```
11: def bar(e,l): #compte le nombre d'élément e dans la liste l
12:     nb=0
13:     for x in l:
14:         if x==e:
15:             nb=nb+1
16:     return nb
17: def foo(e,l): #compte le nombre d'élément e dans la liste l
18:     n=len(l)
19:     if n==0:
110:         return 0
111:     else:
112:         if l[n]==e:
113:             return 1+foo(e,l[:n-1])
114:         else:
115:             return foo(e,l[:n-1])
116:
117: l=['u','n','e',' ','l','i','s','t','e'] " #compter les 'e' avec foo et bar
118: nb=0          #et verifier que c'est identique
119: e='e'
120: if bar(e,l)==foo(e,l):
121:     print 'meme valeur'
```

Rappel : l'expression $l[:p]$ sélectionne tous les éléments de la liste l d'indice inférieur à p .

1. Pour chaque fonction donner ses paramètres effectifs, ses variables locales, les variables globales et celles dont la valeur peut-être modifiée (indiquer les numéros de lignes en cas d'homonymie).
2. L'exécution du script peut-elle déclencher une erreur ?
3. La fonction *foo* est-elle récursive ? Si oui, définir les cas terminaux, les appels récursifs et dire si ceux-ci sont récursifs terminaux. Même question pour la fonction *bar*.
4. Réécrire la fonction $bar(e,l)$ en utilisant une instruction *while* à la place du *for*

Exercice 2 On suppose donnée une fonction *estliste* qui prend un argument x et renvoie *True* si x est une liste et *False* sinon.

1. Ecrire une fonction *nbeltnonliste(l)* qui renvoie le nombre d'éléments e_i de la liste $l = [e_1, \dots, e_n]$ qui ne sont pas une liste et renvoie 0 pour $l = []$.
2. Aplatiser une liste l revient à remplacer tout élément de l qui est une liste par ses éléments et à répéter le processus jusqu'à ce que la liste ne contienne plus de liste comme élément. Exemple aplatiser $[1, [["ab", 3], "cd"], [[4]]]$ donne $[1, "ab", 3, "cd", 4]$.
 - (a) Donner l'aplatissement de la liste $[[["ad"], "ab", 3.0], ["cd", [4]], 2]$.
 - (b) Ecrire une fonction *flatten(l)* qui prend une liste l et renvoie la liste correspondant à la liste l aplatie.

(c) La fonction *flatten* est-elle récursive ?

3. En n'utilisant que la fonction *type* de python qui renvoie le type d'un objet, écrire la fonction *estliste*. Indication, ne pas chercher à utiliser analyser le résultat de la fonction *type* mais utiliser le fait qu'elle renvoie la même valeur pour deux objets de même type.

Exercice 3 Exercice pour les étudiants de M1 TAL.

Un nom est une chaîne de caractères correspondant à un nom par exemple 'paul'. Un verbe est une chaîne de caractère correspondant à un verbe par exemple 'aime'. Un groupe complément est une chaîne de caractère correspondant à un complément d'objet direct par exemple 'les fraises'. La liste *LN* est une liste de noms, la liste *LV* est une liste de verbes, la liste *LC* est une liste de groupes compléments. Ces trois listes seront des variables globales et dans ce qui suit un nom sera simplement un élément de *LN* (et de même pour verbe et groupe complément). La catégorie d'un nom est la lettre 'N', la catégorie d'un verbe est 'V', la catégorie d'un groupe complément est 'C'.

1. Ecrire une fonction *affiche(l)* dont le paramètre formel *l* est une liste d'éléments qui sont des chaînes de caractères, et qui renvoie la chaîne résultant de la concaténation de tous les éléments de *l* séparés par des blancs. Par exemple ['Paul', 'aime', 'les fraises'] s'écrit 'Paul aime les fraises'.
2. Ecrire une fonction *ajoute(l)* spécifiée comme suit :
 - le paramètre formel *l* est une liste dont les éléments sont des listes de la forme [*chaîne, catégorie*] avec catégorie valant 'N', 'V', 'C',
 - pour chaque élément [*chaîne, catégorie*] la fonction ajoute *chaîne* à la liste (*LN* ou *LV* ou *LC*) des éléments de la catégorie *catégorie*.Par exemple *ajoute*([['pierre', 'N'], ['la peinture', 'C']]) ajoute 'pierre' à *LN* et 'la peinture' à *LC*.

Que fait votre fonction si la chaîne appartient déjà à la liste, par exemple si 'pierre' était déjà dans *LN* (ne pas modifier votre fonction, uniquement répondre à la question) ?
3. Ecrire une fonction *analyse1(l)* dont le paramètre formel *l* est une liste de chaînes de caractères et qui renvoie la liste donnant la catégorie de chaque élément de *l*. Par exemple, si *LN* contient 'paul', *LV* contient 'aime' et *LC* contient 'les fraises', alors *analyse1(l)* renvoie ['N', 'V', 'C'] pour l'exemple de la première question. Si un élément de *l* ne correspond pas à un élément des listes *LN, LV, LC*, la fonction renvoie la liste vide.
4. Ecrire une fonction *genere()* qui retourne la liste de toutes les phrases possibles à partir de *LN, LV, LC* dont l'analyse est ['N', 'V', 'C']. Si une des trois listes est vide alors la fonction renvoie la liste vide.

Exercice 4 Exercice pour les étudiants de L1.

La notation $a \equiv b \pmod{n}$ signifie que $a - b$ est divisible par n .

La sécurité informatique repose en partie sur la capacité à fabriquer des grands nombres premiers. Un algorithme utilisé pour savoir si un nombre impair n est premier est le test de Miller-Rabin.

Test de Miller-Rabin : $n - 1$ s'écrit $2^s d$ avec d impair. Si n est premier et si $a \in \{1, 2, \dots, n - 1\}$ alors $a^d \equiv 1 \pmod{n}$ ou $a^{d2^r} \equiv -1 \pmod{n}$ pour un $r \in \{0, 1, \dots, s - 1\}$. Le nombre n passe le test pour une valeur $a \in \{1, \dots, n - 1\}$ tirée au hasard si une des deux propriétés est vérifiée. Si un nombre n réussit le test pour une valeur a tirée au hasard, il n'est pas premier avec une probabilité inférieure à $1/4$ (donc il est premier avec probabilité supérieure à $3/4 = 1 - 1/4$).

1. Ecrire une fonction $decomp(n)$ qui renvoie la liste $[s, d]$ telle que $n - 1 = 2^s d$ si n est impair, sinon elle renvoie $[-1, -1]$.
2. Ecrire une fonction $testMR(n)$ qui renvoie $True$ si et seulement si le test de Miller Rabin réussit pour une valeur a tirée au hasard dans $\{1, \dots, n - 1\}$.
3. Ecrire une fonction $primauté(n, k)$ qui renvoie $True$ si et seulement si le nombre n satisfaisait le test de Miller-Rabin pour k valeurs tirées au hasard. Le nombre n est dit pseudo-premier si $primauté(n, k) = True$ pour une valeur k choisie assez grande.
4. Quelle est la probabilité qu'un nombre n ne soit pas premier si $primauté(n, 5)$ renvoie $True$?
5. Ecrire un script qui calcule la liste des nombres pseudo-premiers de 2 à 100000 en utilisant le test de Miller-Rabin avec $k = 5$ itérations.

Rappel : la fonction $random(l, p)$ renvoie une valeur aléatoire entre l et p .