T NSI Exercices sur la récursivité

Exercice 1 La fonction mystère

Que fait la fonction suivante ?

```
def mystere(L, M = []):
    if not L:
        return M
    a=L.pop(0)
    if a not in M:
        M.append(a)
    return mystere(L,M)
```

Retirer les doublons

Exercice 2 Le retour de la fonction mystère

Que fait la fonction suivante ?

```
def mystere2(L):
    if len(L)==1:
        return L[0]
    if L[0]< L[1]:
        L.pop(1)
    else:
        L.pop(0)
    return mystere2(L)
```

Exercice 3 Calcul du pgcd de deux nombres

Rappel : le **pgcd** de deux entiers naturels a et b est le plus grand diviseur commun à a et b.
Exemple : pgcd(20, 24) = 4
Les diviseurs de 20 sont : 1, 2, **4** , 5, 10, 20.
Les diviseurs de 24 sont : 1, 2, 3, **4**, 6, 8, 12, 24
Le plus grand diviseur qui soit commun aux deux nombres est bien 4.

Ecrire en Python une fonction récursive `pgcd(a , b)` renvoyant le plus grand diviseur commun de deux nombres a et b .
Pour cela, on utilisera le résultat mathématique suivant :
" pgcd(a , b) = pgcd(b, r) où $a = bq + r$ " (dans la division euclidienne de a par b, q est le quotient et r est le reste)

Exercice 4 Nombre d'adhérents

Une association a remarqué que d'une année sur l'autre :
- elle perd 5 % de ses adhérents
- elle gagne 200 nouveaux adhérents

1) Le nombre d'adhérents de cette association était égal à 2000 au 1ier janvier 2019.
a) Montrer que le nombre d'adhérents est égal à 2100 en 2020 et sera égal à 2195 en 2021.
b) Si on note u_n le nombre d'adhérents n années après 2019, exprimer u_{n+1} en fonction de u_n .
c) Ecrire en Python une fonction récursive nommée `nombre(n)` affichant le nombre théorique d'adhérents après n années, $n \geq 1$.

2) En utilisant la fonction `nombre(n)` précédente et une boucle, faire afficher le nombre théorique d'adhérents au cours des 20 prochaines années.

Exercice 5 Division euclidienne

Faire la division euclidienne de deux entiers naturels a et b (b non nul) revient à compter combien de fois on peut ôter b de a (sans que le résultat devienne négatif). Ce nombre de fois est appelé le *quotient* et la quantité restante est appelée le *reste*.
Par exemple, diviser 26 par 6 revient à faire 4 soustractions successives :
26 – 6 = 20
20 – 6 = 14
14 – 6 = 8
8 – 6 = 2 (on s'arrête car 2 < 6)
Ici, le quotient est donc q = 4 et le reste r = 2.

Ecrire en Python une fonction récursive nommée `division(a, b, q=0)` qui retourne le quotient et le reste entiers de la division euclidienne de a par b.

Exercice 6 Etude d'une fonction récursive

On considère le programme suivant :

```
def f(a,b):
    if b==1:
        return a
    return a + f(a, b-1)
```

1) En exécutant le programme à la main, dire quel résultat retourne `f(4,3)`.
2) En déduire la signification de la valeur retournée par cette fonction pour deux entiers naturels non nuls a et b.
3) Pour une fonction récursive, un cas de base est un cas qui ne nécessite pas d'appel récursif à la fonction.
Quel est le cas de base de cette fonction ?
4) Qu'est-ce qui garantit que le programme s'arrête ?
5) La complexité de cette fonction est-elle linéaire ? quadratique ? exponentielle ?

Exercice 7 La fonction compteur

Ecrire une fonction récursive `compteur(chaine,car)` retournant le nombre de fois que l'on peut compter le caractère `car` dans la chaîne `chaine`.
Par exemple, `compteur( "abracadabra", "a")` renvoie le nombre 5 car il y a 5 "a" dans la chaîne "abracadabra".

Exercice 8 Coefficients binomiaux

1) Ecrire une fonction récursive `binomial( n , k )` qui retourne la valeur du coefficient binomial $\binom{n}{k}$ en exploitant la relation de récurrence du triangle de Pascal :

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

Et en sachant, de plus, que pour tout entier naturel n, $\binom{n}{0} = \binom{n}{n} = 1$.

2) Tenter de calculer binomial (30 , 15). Que pensez-vous de l'efficacité de cette fonction ?

3) Améliorer les performances de cette fonction en utilisant la technique de mémoïsation dans un dictionnaire, et en utilisant la propriété $\binom{n}{k} = \binom{n}{n-k}$