D.S. 1, I.P.T. MPSI 1

1 Questions de cours

a) Ecrire une fonction PYTHON qu'on appellera `Somme` qui prend en argument un entier naturel qu'on notera `n` et qui renvoie un flottant représentant la valeur de $\sum_{k=1}^{n} \frac{k}{k^2+1}$

b) Ecrire une fonction PYTHON qu'on appellera `Somme2` qui prend en argument un entier naturel qu'on notera `n` et qui renvoie la somme des entiers impairs $k \in [\![1, n]\!]$ i.e. la valeur de :

$$\sum_{k \text{ impair},\ 1 \leq k \leq n} k.$$

c) Pourquoi le code PYTHON suivant ne marche-t-il pas ? Quel est le message d'erreur affiché ?

```
L=[]
for i in range(10):
    L[i]=i**3
```

d) Changer la première ligne du code précédent pour que le code fasse ce qu'on attend, en créant au départ une liste avec des zéros.

e) Créer *de deux façons différentes*, une liste contenant 100 entiers aléatoires pris entre 1 et 50. On rappelle que la commande `randint(1,50)` renvoie justement un entier aléatoire entre 1 et 50. Cette commande est dans le module `random` qu'il faudra importer dans votre code.

f) Ecrire une fonction python qui prend en argument deux entiers naturels `m` et `n` et renvoie la valeur de $\prod_{i=1}^{m} \prod_{j=1}^{n} (i + 2j)$.

g) On exécute le code suivant :

```
def essai(a):
    a=a+1
    return a
a=2
print(essai(a))
print(a)
```

Qu'affichent les deux derniers `print` ? Expliquer ?

h) On exécute le code suivant :

```
def essai2(L):
    L[1]=2
    return L
L=[0,1]
essai2(L)
print(L)
```

Qu'affiche le dernier `print` ? Expliquer.

2 Mots dans un texte

a) Ecrire une fonction `EsTuLa(objet,L)`, où `L` est une liste, qui teste s'il existe une entrée `L[i]` de `L` telle que `L[i]` soit égale à l'argument `objet` : cette fonction renverra un booléen.

N.B. On n'autorise pas l'utilisation de la fonction `in` qui fait exactement cela en python, car on veut bien comprendre la complexité de ce test.

b) Ecrire une fonction `decompose` qui prend en argument une chaîne de caractères représentant un texte en français qu'on appellera `texte` et renvoie une liste dont les entrées sont des chaînes de caractères formées des différents mots de `texte`.

Par exemple avec ; `texte="un petit essai pour voir"`, `decompose(texte)` renverra :
`['un', 'petit', 'essai', 'pour', 'voir']`

c) Avec les mêmes hypothèses sur `texte` que dans la question précédente, on veut écrire une fonction `Recherche` qui prend deux arguments `mot,texte` qui sont des chaînes de caractères et qui renvoie `True` si `mot` est dans `texte` et `False` sinon.

On propose deux méthodes :

(i) Avec `EsTuLa` et `decompose`.

(ii) Puis sans utiliser `decompose` ni `EsTuLa` mais en travaillant directement sur la chaîne de caractères `texte` et en s'arrêtant dès qu'on trouve le mot, ce qui est beaucoup plus économique.

d) Dans une liste comme celle renvoyée par la fonction `decompose`, on peut avoir bien sûr des répétitions du même mot. Par exemple avec

`texte="le lutin bleu est bleu comme le ciel car on peut démontrer que le ciel est bleu"`, `decompose(texte)` donnera :

`['le', 'lutin', 'bleu', 'est', 'bleu', 'comme', 'le', 'ciel', 'car', 'on', 'peut', 'démontrer', 'que', 'le', 'ciel', 'est', 'bleu']` ;

On veut pouvoir éliminer les répétitions d'une telle liste i.e. écrire une fonction `elimineRepet` qui prend comme argument une liste `L` et renvoie une autre liste qui contient les mêmes entrées, mais en enlevant les répétitions comme suit : `['le', 'lutin', 'bleu', 'est', 'comme', 'ciel', 'car', 'on', 'peut', 'démontrer', 'que']`

(i) Une méthode très rapide consiste à passer par le type ensemble (même si la fonction renvoie une liste à la fin). La commande `set` transforme une liste ou un tuple en ensemble (avec des accolades) où l'ordre ne compte pas et les répétitions sont supprimées.

Sauriez-vous écrire une fonction `elimineRepet1` avec cette idée ?

(ii) La méthode précédente a ses inconvénients. Le premier est que le type ensemble ne gère pas l'ordre et que l'ordre des éléments n'est pas respecté. Par exemple, elle peut renvoyer : `['que', 'bleu', 'car', 'est', 'lutin', 'le', 'ciel', 'on', 'peut', 'comme', 'démontrer']`

Ce n'est pas forcément grave suivant ce qu'on veut faire bien sûr.

Le second, plus grave, est qu'on ne comprend pas la complexité (le temps de calcul) des opérations faites.

Ecrire une fonction `elimineRepet2` qui n'utilise que le type liste et des opérations élémentaires sur les listes ou une fonction déjà écrite.

e) Ecrire une fonction `ListeNbreOcccurences` qui prend en argument une chaîne de caractères qu'on appellera `texte` et renvoie une liste de couples où chaque couple est formé d'un mot du texte et de son nombre d'occurrences dans `texte`. On évitera les répétitions dans la liste.

Ainsi avec la variable `texte` précédente, `ListeNbreOccurences(texte)` renverra :

```
[['le', 3], ['lutin', 1], ['bleu', 3], ['est', 2], ['comme', 1], ['ciel', 2], ['car', 1],
 ['on', 1], ['peut', 1], ['démontrer', 1], ['que', 1]]
```

f) Cas d'un texte sans espaces. Par exemple `texte="lebonbonbon"`. Dans ce texte on dira que le mot `"bonbon"` apparaît deux fois (la première fois à en commençant à `texte[2]` la seconde fois à `texte[6]`).

(i) Ecrire une fonction `occurences(mot,texte)` qui renvoie le nombre d'occurrences de `mot` dans `texte` (et renvoie 0 si le mot n'est pas dans le texte).

(ii) Ecrire une fonction `afficheMot2` qui reçoit en argument une chaîne de caractères `texte` sans espaces, et renvoie une liste donnant tous les mots de deux lettres (qui n'ont pas forcément de sens en français) dans `texte` avec leur nombre d'occurrences.

Autrement dit pour `texte="lebonbonbon"`, si `afficheMot2(texte)` renvoie la liste sans répétitions, on obtiendra :

`[['le', 1], ['eb', 1], ['bo', 3], ['on', 3], ['nb', 2]]`

Vous pouvez réfléchir, (éventuellement à la maison), à différentes idées pour optimiser cette fonction...

Culturel : ces problèmes de recherche de mot dans un texte sont cruciaux pour les navigateurs internet. Vous vous doutez bien qu'il existe, pour ce faire, des algorithmes moins naïfs que ceux des questions précédentes. Ils reposent sur des tris. Nous y reviendrons en D.M...