Contrôle continu - Programmation (MASCO 1)
Durée : 2h
Documents autorisés : le polycopié du cours

A lire avant de commencer

— Certaines fonctions que l'on vous demande d'écrire existent déjà en Python. Si c'est le cas, vous n'avez **pas le droit** de les utiliser dans votre solution. L'idée est de comprendre comment ces fonctions ont été programmées ! En particulier vous ne devez pas utiliser les fonctions de recherche et de remplacement dans des chaînes de caractères.

— Ecrivez de manière la plus lisible possible, les copies illisibles ne seront pas lues.

1 Remplacement dans une chaîne de caractères

Q1.1 Ecrire une fonction `remplace1(s, x, y)` qui prend en argument une chaîne de caractères `s` et deux caractères `x` et `y` et renvoie une nouvelle chaîne de caractères correspondant à `s` dans laquelle toute occurrence du caractère correspondant à `x` est remplacée par le caractère correspondant à `y`. Exemple :

```
print(remplace1('ababb', 'a', 'A'))
AbAbb
```

Q1.2 Ecrire une fonction `remplace2(s, l, y)` qui prend en argument une chaîne de caractères `s`, une liste de caractères `l` et un caractère `y` et renvoie une nouvelle chaîne de caractères correspondant à `s` dans laquelle toute occurrence d'un caractère de la liste `l` est remplacée par le caractère `y`. Exemple :

```
print(remplace2('abcabb', ['a', 'b'], 'X'))
XXcXXX
```

Q1.3 Ecrire une fonction `remplace3(s, d)` qui prend en argument une chaîne de caractères `s` ainsi qu'un dictionnaire `d` et renvoie une nouvelle chaîne de caractère correspondant à `s` dans laquelle toute occurrence d'un caractère qui constitue une clef de `d` est remplacée par la valeur associée à cette clef. Exemple :

```
print(remplace3("abacada", {'a':'1', 'b':'2'}))
121c1d1
```

Q1.4 Ecrire une fonction `remplace4(s, lx, ly)` qui prend en argument une chaîne de caractères `s` ainsi que deux listes de caractères `lx` et `ly` et renvoie une nouvelle chaîne de caractère correspondant à `s` dans laquelle toute occurrence d'un caractère appartenant à `lx` est remplacé par un caractère se trouvant dans `ly` au même indice. Exemple :

```
print(remplace4("abacada", ['a', 'b'], ['1', 'b']))
121c1d1
```

2 n-grammes et mesure cosinus

Q2.1 Ecrire une fonction `unigrammes(s)` qui prend en argument une chaîne de caractères `s` et renvoie un dictionnaire contenant le nombre d'occurrences de chaque caractère de `s`

```
d = unigrammes("abacadab")
print(list(d.items()))
[('a', 4), ('b', 2), ('c', 1), ('d', 1)]
```

Q2.2 Ecrire une fonction `ngrammes(s,n)` qui prend en argument une chaîne de caractères `s` et renvoie un dictionnaire contenant le nombre d'occurrences de tous les n-grammes de `s`. Un n-gram est une séquence de n lettres.

```
d = ngrammes("abacadab", 2)
print(list(d.items()))
[('ab', 2), ('ba', 1), ('ac', 1), ('ca', 1), ('ad', 1), ('da', 1)]
```

Q2.3 Ecrire la fonction `produitScalaire(v1, v2)` qui prend en argument deux vecteurs `v1` et `v2` de même dimension, représentés par deux listes de nombres décimaux, et renvoie le produit scalaire des deux vecteurs. On rappelle que le produit scalaire de deux vecteurs v_1 et v_2 est défini de la façon suivante

$$v_1.v_2 = \Sigma_{i=1}^{d} v_1[i] \times v_2[i]$$

où d est la dimension des vecteurs v_1 et v_2. Exemple :

```
print(produit_scalaire([1,2,3],[4,5,6]))
32
```

Q2.4 Ecrire la fonction `cosinus(v1, v2)` prend en entrée deux vecteurs `v1` et `v2` et renvoie la mesure cosinus des deux vecteurs. Cette dernière est calculée de la manière suivante :

$$cosinus(v_1, v_2) = \frac{v_1.v_2}{||v_1|| \times ||v_2||}$$

où $||v||$ est la norme du vecteur v.
Cette dernière peut être calculée à l'aide du produit scalaire de la manière suivante :

$$||v|| = \sqrt{v.v}$$

en d'autres termes la racine carrée du produit scalaire de v par lui-même. Note : la fonction racine carrée de Python est définie dans le module `math`. Elle s'appelle `sqrt`.

```
print(cosinus([1,0,1],[1,1,0]))
1
```

Q2.5 Ecrire une fonction `dic2vec(d1, d2)` qui prend en argument deux dictionnaires `d1` et `d2` et renvoie deux listes `l1` et `l2`. `l1` et `l2` sont de même taille, qui est la taille de l'union des clefs de `d1` et de `d2`. La liste `l1` correspond aux valeurs du dictionnaire `d1` et la liste `l2`, aux valeurs du dictionnaire `d2`. Cependant, si une clef `c` de `d2` n'apparaît pas dans `d1`, alors, on aura à la place correspondant à `c` dans `l1` la valeur `0`. Exemple : si `d1 = {'aa':1, 'bb':2}` et `d2 = {'aa':1, 'cc':2}`, on aura comme sortie `l1 = [1, 2, 0]` et `l2 = [1, 0, 2]`. `l1` et `l2` correspondent au nombre de `aa`, de `bb` et de `cc` respectivement dans `d1` et `d2`. Attention : l'ordre dans lequel apparaîssent les valeurs dans `l1` et `l2` n'est pas important, ce qui est important c'est que cet ordre soit le même dans les deux listes : si le nombre de `bb`, par exemple, apparaît en position 2 dans `l1` alors il doit aussi apparaître en position 2 dans `l2`.

Q2.6 Ecrire la fonction `cosinusString(s1, s2, n)` qui prend en argument deux chaînes de caractères `s1` et `s2` ainsi qu'un entier `n`. La fonction calcule le nombre d'occurrences des n-grammes de `s1` et `s2` et renvoie le cosinus des vecteurs de comptes des n-grammes.