ENSAE mardi 3 décembre 2013

Cette interrogation écrite compte pour 5 points ajoutés à l'ensemble des notes de la matière. Tous les documents sont autorisés. La durée est d'une demi-heure. Vous devrez imprimer le résultat en n'omettant pas d'y ajouter votre nom et le numéro de l'énoncé qui vous aura été distribué.

1

1) Ecrire une fonction qui remplace les caractères accentués par la lettre correspondante sans accent ? On pourra se limiter à la correction de trois lettres accentuées. (2 points)

Rappel : pour un programme contenant des accents, il faut ajouter `#coding:latin-1`.

2) Que fait l'instruction `"".join(sorted(s))` ? (où `s` est une chaîne de caractères) (1 point)

3) Ecrire une fonction qui détecte si deux mots sont des anagrammes ? (2 points) (exemple : décentre et cédèrent).

Correction

1) Quelques solutions :

```
def remplace_accent(s):
    return s.replace("é","e").replace("à","a").replace("ù","u")

def remplace_accent(s):
    s = s.replace("é","e")
    s = s.replace("à","a")
    s = s.replace("ù","u")
    return s

def remplace_accent(s):
    for a,b in [('à','a'), ('é','e'), ('ù','u')]:
        s = s.replace(a,b)
    return s
```

Il est possible de faire sans la fonction `replace` :

```
def remplace_accent(mot):
    liste = list(mot)
    for i in range(len(liste)) :
        if liste[i] == "é" : liste[i] = "e"
        elif liste[i] == "è" : liste[i] = "e"
        elif liste[i] == "à" : liste[i] = "a"
    return "".join(liste)
```

Si on ne convertit pas la chaîne de caractères en liste, en utilisant la fonction suivante par exemple :

```
def remplace_accent(mot):
    for i in range(len(mot)) :
        if mot[i] == "é" : mot[i] = "e"
        elif mot[i] == "è" : mot[i] = "e"
        elif mot[i] == "à" : mot[i] = "a"
    return "".join(mot)
```

On provoque l'erreur suivante :

```
  File "interro_rapide_30_minutes_2013_12.py", line 14, in remplace_accent
    if mot[i] == "é" : mot[i] = "e"
TypeError: 'str' object does not support item assignment
```

Les chaînes de caractères ne sont pas modifiables : on ne peut pas remplacer un caractère par un autre.

2) L'instruction "".`join`(`sorted`(`s`)) trie les lettres d'un mot par ordre alphabétique. Autrement dit, le résultat est un des anagrammes de `s`.

3) Deux mots sont des anagrammes s'ils sont composés des mêmes lettres. Pour se faire, on peut s'assurer que les lettres triées dans l'ordre alphabétique sont les mêmes pour les deux mots :

```
def anagrammes(mot1, mot2):
    mot1 = remplace_accent(mot1)
    mot2 = remplace_accent(mot2)
    return "".join(sorted(mot1)) == "".join(sorted(mot2))
```

ENSAE mardi 3 décembre 2013

Cette interrogation écrite compte pour 5 points ajoutés à l'ensemble des notes de la matière. Tous les documents sont autorisés. La durée est d'une demi-heure. Vous devrez imprimer le résultat en n'omettant pas d'y ajouter votre nom et le numéro de l'énoncé qui vous aura été distribué.

2

1) Ecrire une fonction qui compte la fréquence des lettres dans un mot. (1 point)

2) Ecrire une fonction qui détermine si une chaîne de caractères contient les lettres d'un même mot en double exemplaire : *feraiifare* contient deux fois le mot *faire*. (2 points)

```
def est_mot_double(s) :
    ...
    return True ou False

print (est_mot_double('feraiifare'))  # affiche True
```

Astuce : que vérifient les fréquences de chaque lettre pour une telle chaîne de caractères ?

3) Ecrire une fonction qui prend deux mots de même longueur et intercale leurs lettres comme suit : *ferai* et *faire* donne *fFeArIaRiE*. (2 points)

Correction

1) Compter la fréquences des lettres dans un mot est assez simple à faire avec un dictionnaire :

```
def frequence_lettre(mot) :
    res = { }
    for c in mot :
        res[c] = res.get(c,0)+1
    return res
```

2) Si un mot est en double exemplaire alors les fréquences de ses lettres sont toutes paires. Voici plusieurs versions équivalentes, de la plus longue à la plus condensée :

```
def est_double_mot(mot):
    freq = frequence_lettre(mot)
    for f in freq :
        if freq[f] % 2 != 0 :
            return False
    return True

def est_double_mot(mot):
    freq = frequence_lettre(mot)
    reste = [ v%2 for k,v in freq.items() ]
    return sum(reste) == 0

def est_double_mot(mot):
    return sum( [ v%2 for k,v in  frequence_lettre(mot).items() ] ) == 0
```

3)

```
def intercale(mot1,mot2):
    res = ""
    for i in range(len(mot1)) :
        res += mot1[i] + mot2[i]
    return res
```