

CORRIGÉ DU TP11 — 22 JANVIER 2022

EXERCICE 1 — Résumé de texte en 200 mots +/- 10%...

PARTIE A — RECHERCHE DE CARACTÈRES, ET DE CHAÎNES DE CARACTÈRES

On se propose dans un premier temps de coder quelques fonction “simples” sur les chaînes de caractères.

1. Proposer une fonction `test(résumé, caractère)` prenant en argument une chaîne de caractères appelée `résumé` et un caractère nommé `caractère`, et retournant `True` si ce caractère est présent dans le résumé.

```
1 def test(résumé, caractère):
2     return (caractère in résumé)
```

2. On souhaite créer une fonction `positioncaractère` prenant en argument une chaîne de caractère appelée `résumé` et un caractère, et renvoyant la liste des positions où apparaît le caractère dans le résumé.

Par exemple : `positioncaractère('Quel beau texte', 'u')` doit retourner `[1,8]`.

Compléter le code ci-dessous :

```
1 def positioncaractère (résumé, carac):
2     ListPos = []
3     for i in range(len(résumé)):
4         if résumé[i] == carac: # Ligne à compléter
5             ListPos = ListPos + [i] # Ligne à compléter
6     return ListPos
```

3. En s'inspirant des fonctions précédentes, proposer une fonction `nombremots` prenant en argument une chaîne de caractères appelée `résumé` et renvoyant le nombre de mots de cette chaîne de caractères (pour simplifier, on pourra supposer que la chaîne de caractères `résumé` ne contient pas d'apostrophe).

```
1 def nombremots(résumé):
2     nbmots = 0
3     for i in range(len(résumé)):
4         if résumé[i] == ' ':
5             nbmots = nbmots + 1
6     return nbmots + 1
```

Monsieur V a donné comme consigne aux élèves d'utiliser au moins une fois le mot “enfance” dans leur résumé. On propose ici de coder une fonction permettant de vérifier que la consigne a été respectée.

On souhaite tout d'abord coder une fonction `appartient1(résumé,mot)` prenant en argument une chaîne de caractère appelée `résumé` et un mot de deux lettres et qui retourne `True` si le mot apparaît bien dans le texte.

4. Parmi les 3 fonctions ci-dessous, quelle(s) sont la (les) fonction(s) pouvant convenir ? Pour les fonctions non valides, préciser les problèmes identifiés.

```
1 # PROPOSITION 1
2 def appartient1(résumé,mot) :
3     result = False
4     if mot in résumé :
5         result = True
6     return result
```

La proposition 1 ne convient pas. En effet, si le mot recherché est “un”, et que le résumé contient le mot “dune”, la fonction renverra à tort la valeur `True`.

```

1  # PROPOSITION 2
2  def appartient1(résumé,mot) :
3      result = False
4      if mot[0] in résumé :
5          for j in positioncaractère (résumé,mot[0]):
6              if j ==0:
7                  if mot ==résumé[0]+résumé[1] and résumé[2] in '!.?! ;:':
8                      result = True
9              if j >0:
10                 if mot[1] ==résumé[j+1] and résumé[j-1] == " " and résumé[j+2] in '!.?! ;:':
11                     result = True
12     return result

```

La proposition 2 convient.

```

1  # PROPOSITION 3
2  def appartient1(résumé,mot) :
3      result = True
4      if len( positioncaractère (résumé,mot[0])) ==len( positioncaractère (résumé,mot[1])) :
5          for i in range(len( positioncaractère (résumé,mot[0]))) :
6              if positioncaractère (résumé,mot[0])[1] != positioncaractère (résumé,mot[0])[0] +1 :
7                  result = False
8      else :
9          result = False
10     return result

```

La proposition 3 ne convient pas, à cause de l'obscur ligne 6, qui compare les deux premières positions de l'initiale du mot recherché... Au passage, ce code renverra un joli message d'erreur (style "index out of range") lorsque les deux caractères du mot recherché apparaissent une seule fois dans la chaîne résumé.

PARTIE B — RECHERCHE DE PLAGIAT

Parmi les algorithmes permettant de détecter les plagiat on peut citer *l'algorithme de Levenshtein*. On appelle *distance de Levenshtein* entre deux mots M et P le coût minimal pour transformer M en P en effectuant les seules opérations élémentaires suivantes : substitution d'un caractère de M par un caractère différent de P, insertion (ou ajout) dans M d'un caractère de P, suppression (ou effacement) d'un caractère de M.

On associe à chacune de ces opérations un coût. Généralement, le coût est égal à 1 pour chacune des trois opérations.

5. Pour deux mots M et P donnés, comment interpréter une distance de Levenshtein proche de 0 ?

L'interprétation naïve d'une distance de Levenshtein proche de 0 est que les mots M et P sont assez "ressemblants". Plus généralement, le fait que la distance de Levenshtein entre deux textes soit proche de zéro est un "indicateur de plagiat" (assez peu fiable au premier abord...).

6. On propose de coder une fonction calculant la *distance de Levenshtein simplifiée* entre deux mots de même longueur, et uniquement basée sur l'opération de substitution : quand on compare deux mots M et P, la distance augmente d'une unité dès que l'on repère qu'il suffit de substituer un caractère de P pour retrouver le mot M.

Exemple : M : "bonjour" et P : "bongour" la distance de Levenshtein simplifiée est égale à 1.

Ecrire une fonction **distance** prenant en argument deux chaînes de caractères nommées **résumé1** et **résumé2** de même longueur et qui retourne la distance de Levenshtein simplifiée.

```

1  def distance (mot1, mot2):
2      if len(mot1) !=len(mot2):
3          return 'Distance non définie , chaînes de longueurs distinctes '
4      else :
5          dist =0
6          for k in range(len(mot1)):
7              if mot1[k] !=mot2[k]:
8                  dist = dist +1
9     return dist

```

PARTIE C — PROGRAMME PRINCIPAL

7. Proposer un programme principal effectuant les opérations suivantes.

- Demande à deux élèves de saisir leurs résumés que l'on stockera dans les variables `résumé1` et `résumé2` (on supposera que ces deux résumés possèdent la même longueur) ;
- Décompte des mots et vérification que la consigne 200 mots \pm 10% est respectée. Affichage d'un message précisant si cette consigne a été respectée ou pas.
- Vérification pour les deux résumés que le mot "enfance" apparaît au moins une fois dans le texte. Affichage d'un message précisant si la consigne a été respectée ou pas. A cet effet, on pourra supposer qu'une fonction `appartient2` a été codée, qui prend en argument une chaîne de caractère appelée `résumé` et un mot quelconque, et qui précise si ce mot apparaît dans le résumé ainsi que le nombre de fois où il apparaît.
- Calcul de la distance simplifiée de Levenshtein et affichage d'un message d'alerte si cette distance franchit la valeur seuil 50.

```

1  # PROGRAMME PRINCIPAL
2
3  L = []
4  for k in range(2):
5      print(' Veuillez saisir votre résumé, utilisateur ',k+1)
6      L = L + [ input() ]
7  for k in range(2):
8      if abs(nombremots(L[k]) - 200) > 20:
9          print('Nombre de mots incorrect, utilisateur ',k+1)
10 for k in range(2):
11     if appartient2(L[k], 'enfance')[1]>0:
12         print('Bravo! Vous avez utilisé le mot enfance, utilisateur ',k+1)
13     else:
14         print('Oooooohhhh! Vous n avez pas utilisé le mot enfance, utilisateur ',k+1)

```

PARTIE D — POST-TRAITEMENT

Sensible à la syntaxe et à la grammaire, Monsieur V souhaite corriger une erreur courante, celle consistant à débiter une phrase par une lettre minuscule. A cette fin, il cherche à écrire un programme repérant ces erreurs, et les rectifiant de manière systématique.

Pour réaliser ce programme, il peut être utile de savoir qu'en informatique, à chaque caractère est associé un nombre appelé **code ASCII**, qui est un entier compris entre 0 et 255.

En outre, la fonction `ord` permet d'obtenir le code ASCII d'un caractère. Sa bijection réciproque est la fonction `chr`, qui permet d'associer à un entier de $\llbracket 0, 255 \rrbracket$ le caractère ayant cet entier pour code ASCII.

Par exemple : `ord('A')=65`, et `chr(65)='A'`.

8. Ecrire une fonction `MAJ(carac)`, qui reçoit comme paramètre un caractère `carac`, qui retourne la lettre majuscule correspondante lorsque `carac` est une lettre minuscule de l'alphabet, et qui retourne `carac` sinon.

```

1  def MAJ(ch):
2      alf = 'abcdefghijklmnopqrstuvwxyz'
3      if ch in alf:
4          ch = chr(ord(ch)-32)
5      return ch

```

9. Ecrire une fonction `MAJCORR(résumé)`, qui reçoit comme paramètre une chaîne de caractères nommée `résumé`, qui retourne la chaîne de caractères où les erreurs évoquées en préambule de cette partie ont été corrigées.

```
1 def MAJCORR(txt):
2     txt2 = ''
3     long = len(txt)
4     txt2 = txt2 + MAJ(txt[0])
5     k = 1
6     while k < long-2:
7         if txt[k] in '?!':
8             txt2 = txt2 + txt[k] + ' ' + MAJ(txt[k+2])
9             k = k+3
10        else:
11            txt2 = txt2 + txt[k]
12            k = k+1
13    if len(txt2) < len(txt):
14        delta = len(txt) - len(txt2)
15        for k in range(delta):
16            txt2 = txt2 + txt[long-delta+k]
17    return txt2
```