

## DS D'INFORMATIQUE

**Exercice 1.** Le but de cette exercice est d'écrire quelques fonctions permettant d'étudier un texte ou plusieurs textes.

Dans l'ensemble de l'exercice, les textes `ch` considérés seront des chaînes de caractères en majuscule ne contenant pas de caractères spéciaux ni ponctuation. C'est-à-dire des textes du type suivant :

`"BONJOUR A TOI CA VA BIEN AUJOURD HUI"`

Dans cet exemple `AUJOURD` et `HUI` seront considérés comme deux mots distincts.

- Expliquer ce que fait la fonction ci-contre où la
- (1) chaîne de caractères entre guillemets ne contient qu'un espace.

```
1 def mystere(ch) :
2     i = 0
3     while ch[i] != " " :
4         i += 1
5     return ch[:i]
```

- (2) Parmi les quatre fonctions suivantes, déterminer l'unique fonction `egale` qui prend en argument deux chaînes de caractères `ch1` et `ch2` de même longueur (on ne vérifiera pas cette hypothèse) et qui renvoie `True` si les deux chaînes sont égales, et `False` sinon.

**Aucune justification n'est attendue.**

```
1 def egale1(ch1, ch2) :
2     n = len(ch1)
3     for i in range(n) :
4         if ch1[i] != ch2[i] :
5             return False
6     else :
7         return True
```

```
1 def egale2(ch1, ch2) :
2     n = len(ch1)
3     for i in range(n) :
4         for j in range(n) :
5             if ch1[i] != ch2[j] :
6                 return False
7     return True
```

```
1 def egale3(ch1, ch2) :
2     n = len(ch1)
3     i = 0
4     while i < n :
5         if ch1[i] == ch2[i] :
6             return True
7         i += 1
8     return False
```

```
1 def egale4(ch1, ch2) :
2     n = len(ch1)
3     i = 0
4     while i < n :
5         if ch1[i] != ch2[i] :
6             return False
7         i += 1
8     return True
```

- (3) Écrire une fonction `indcarac(ch, caract)` qui prend en argument un texte et un caractère et renvoie la liste des indices du texte où se situe ce caractère. On utilisera pas la méthode `.index`.

Par exemple, `indcarac("BONJOUR A TOI", "O")` renverra `[1, 4, 11]`

Recopier et compléter la fonction `listemots(ch)` ci-contre qui prend en argument un texte pour qu'elle renvoie la liste des mots de ce texte.

- (4) Par exemple,  
`listemots("CA VA BIEN AUJOURD HUI")`  
renvoie  
`["CA", "VA", "BIEN", "AUJOURD", "HUI"]`

```
1 def listemots(ch) :
2     I = indcarac(ch,...)
3     L = [mystere(ch)]
4     for i in range(len(I)-1) :
5         L.append(...)
6     L.append(...)
7     return ...
```

- (5) Écrire une fonction `nbmots(ch)` qui renvoie le nombre de mot de ce texte.  
Par exemple, `nbmots("BONJOUR A TOI CA VA BIEN AUJOURD HUI")` renverra 8.
- (6) Écrire une fonction `finS(ch)` qui renvoie le nombre de mot qui se finisse par un 'S'.  
Par exemple, `finS("JE SUIS SUR DE MOI")` renverra 1.

- (7) Écrire une fonction `maxmot(ch)` qui renvoie le mot le plus long du texte (s'il y en a plusieurs différent de longueur maximal, elle renverra le premier qui apparaît dans le texte). On utilisera pas la fonction `max`.

Par exemple, `fmaxmot("JE SUIS SANS LE SOUS")` renverra "SUIS".

- (8) La fonction `ord` prend en argument une chaîne de caractère et renvoie un entier de la manière suivante :

`ord('A')` renvoie 65, `ord('B')` renvoie 66, `ord('C')` renvoie 67...

Écrire une fonction `effectif(ch)` qui renvoie une liste `F` de taille 26 telle que `F[0]` contient la fréquence d'apparition du `A` dans le texte, `F[1]` celle du `B` et ainsi de suite. On oubliera pas de tenir compte des espaces.

- (9) On considère deux listes  $F = [f_1, \dots, f_n]$  et  $E = [e_1, \dots, e_n]$ . On définit la distance entre ces deux listes de flottant par  $d = \sum_{k=1}^n |f_k - e_k|$ .

Écrire une fonction `dist` qui prend en argument deux listes de flottant de même taille `F` et `E` et qui calcul la distance entre ces deux listes.

Indication : On rappelle la fonction `abs` permet de calculer la valeur absolue d'un flottant.

- (10) On considère qu'il est raisonnable que deux textes soient écrits dans la même langue si la distance entre les deux fréquences d'apparition des lettres dans chaque texte est inférieur ou égale à 30.

Écrire une fonction `langue` qui prend en argument deux textes `ch1` et `ch2` et qui renvoie `True` s'il est probable que les deux textes soient écrits dans la même langue et `False` sinon.