DS D'INFORMATIQUE

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

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

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

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

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

```
def mystere(ch) :
    i = 0
    while ch[i] != "␣" :
        i += 1
    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.

```
def egale1(ch1, ch2) :
    n = len(ch1)
    for i in range(n) :
        if ch1[i] != ch2[i] :
            return False
        else :
            return True
```

```
def egale2(ch1, ch2) :
    n = len(ch1)
    for i in range(n) :
        for j in range(n) :
            if ch1[i] != ch2[j] :
                return False
    return True
```

```
def egale3(ch1, ch2) :
    n = len(ch1)
    i = 0
    while i < n :
        if ch1[i] == ch2[i] :
            return True
        i += 1
    return False
```

```
def egale4(ch1, ch2) :
    n = len(ch1)
    i = 0
    while i < n :
        if ch1[i] != ch2[i] :
            return False
        i += 1
    return True
```

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

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

(4) 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.

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

```
def listemots(ch) :
    I = indcarac(ch,...)
    L = [mystere(ch)]
    for i in range(len(I)-1) :
        L.append(...)
    L.append(...)
    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 apparait 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 chaine de caratère 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 $F = [f_1, ..., f_n]$ et $E = [e_1, ..., e_n]$. 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.

Correction de l'exercice 1. (1) Elle renvoie le premier mot du texte.

(2) La fonction correcte est `egale4`.

(3)

```
def indcarac(ch, carac) :
    I = []
    n = len(ch)
    for i in range(len(ch)) :
        if ch[i] == carac :
            I.append(i)
    return I
```

(4)

```
def listemots(ch) :
    I = indcarac(ch,"␣")
    L = [mystere(ch)]
    for i in range(len(I)-1) :
        L.append(ch[I[i]+1:I[i+1]])
    L.append(ch[I[-1]+1:])
    return L
```

(5)

```
def nbmots(ch) :
    return len(listemots(ch))
```

ou

```
def nbmots(ch) :
    return len(indcarac(ch,"␣"))+1
```

(6)

```
def finS(ch) :
    L = listemots(ch)
    c = 0
    for mot in L :
        if mot[-1] == "S" :
            c += 1
    return c
```

(7)

```
def maxmot(ch) :
    L = listemots(ch)
    motm = L[0]
    for mot in L :
        if len(motm) < len(mot) :
            motm = mot
    return motm
```

(8)

```
def effectif(ch) :
    F = [0]*26
    for e in ch :
        if e != "␣" :
            F[ord(e)-65] += 1
    l = len(indcarac("␣"))
    for i in range(len(F)) :
        F[i] = F[i]/(len(ch)-l)
    return F
```

(9)

```
def dist(F, E) :
    S = 0
    for i in range(len(F)) :
        S += abs(F[i]-E[i])
    return S
```

(10)

```
def langue(ch1, ch2) :
    F, E = effectif(ch1), effectif(ch2)
    if dist(F, E) > 30 :
        return False
    return True
```