TD 8 -
Recherche de chaînes de caractères.

Informatique
MPSI/PCSI - Lycée Thiers

Exercice 1 - Enoncé

Exercice 1. *Recherche de sous-mot*

Ecrire une fonction `recherche(chaine,mot)` prenant en paramètres deux chaines de caractère `chaine` et `mot` et qui retourne `True` ou `False` selon que `mot` apparaisse comme sous-chaine dans `chaine` ou non.

1. A l'aide d'un slicing.
2. Sans utiliser de slicing.
3. Ecrire une fonction `indice(chaine,mot)` qui renvoie `-1` si `chaine` ne contient pas `mot` et qui sinon renvoie l'indice où la première occurence du mot a été trouvée.

Exercice 1 - Correction

1) Avec slicing :

```
def cherche(chaine,mot):
    N = len(chaine)
    n = len(mot)
    for i in range(N-n+1):
        if mot == chaine[i:i+n]:
            return True
    return False
```

2) Sans slicing :

```
def cherche2(chaine,mot):
    N = len(chaine)
    n = len(mot)
    for i in range(N-n+1):
        k = 0
        while k<n and mot[k] == chaine[i+k] :
                k+=1
        if k == n:
            return True
    return False
```

3) En renvoyant l'indice (-1 si mot non trouvé) :

```
def indice(chaine,mot):
    N = len(chaine)
    n = len(mot)
    for i in range(N-n+1):
        k = 0
        while k<n and chaine[i+k]==mot[k]:
            k+=1
        if k==n:
            return i
    return -1
```

Exercice 2 - Application

Exercice 2. *Application.*

Un fichier au format `PDF` est un fichier texte commençant par la chaine de caractère `%PDF` et finissant par `%%EOF` (il peut y avoir d'autres caractères avant et après, qui seront ignorés par un lecteur PDF).

1. Saisir le code suivant :

```
import os
os.getcwd()
```

Il donne le chemin d'accès du dossier ou doit être placé un fichier pour le lire en `python`.

Si on le souhaite, on peut changer ce dossier à l'aide de la fonction `os.chdir` en lui passant en paramètre une chaine de caractère décrivant le chemin d'accès du dossier à utiliser.

Obtenir le "dossier courant" à l'aide des commandes ci-dessus. Y placer (ou y trouver) un (ou plusieurs) fichier PDF.

Exercice 2 - Application

On obtient le contenu du fichier PDF, sauvegardé dans une variable `texte` de type chaine de caractère, à l'aide des commandes suivantes :

```
fichier = open('nom_du_fichier', 'r', errors="surrogateescape")
texte = fichier.read()
fichier.close()
```

2. A l'aide d'une fonction de l'exercice 1, vérifier que le fichier concerné est bien au format PDF.

Exercice 2 - Application

La plupart du temps, un fichier PDF contient une chaine de caractère qui permet de savoir quand il a été créé. C'est une sous-chaine de la forme :

`/CreationDate(D:aaaammjjhhmmss+XX'00')`

où `aaaa` : année, `mm` : mois, `jj` : jour, `hh` : heure, `mm` : minutes, `ss` : secondes, `+XX` décalage GMT. Par exemple :

`/CreationDate(D:20150611214310+01'00')`

décrit une fichier créé le 11 juin 2016, à 21h43m10s, à la longitude du méridien de Paris (+01).

3. Ecrire une fonction `datePDF` qui prend en paramètre le nom d'un fichier, teste si il s'agit d'un PDF, et si possible écrit à l'écran, dates et heures de création (avec décalage GMT). Attention `/CreationDate` et `(...)` peuvent être séparés d'un ou plusieurs espaces.

Exercice 2 - Corrigé

2.

```
f = open(fichier,'r', errors="surrogateescape")
texte = f.read()
f.close()
if -1 < indice(texte,'%PDF') < indice(texte,'%%EOF') :
    print("Le fichier est au format PDF")
else:
    print("Ce n'est pas un PDF")
```

Exercice 2 - Corrigé

3)

```
def datePDF(fichier):
    f = open(fichier,'r', errors="surrogateescape")
    texte = f.read()
    f.close()
    if not(-1<indice(texte,'%PDF')<indice(texte,'%%EOF')):
        print("Ce n'est pas un PDF")
        return
    i = indice(texte,'/CreationDate')
    if i == -1:
        print("Fichier PDF. Date de création introuvable")
        return
    # Suite page suivante
```

Exercice 2 - Corrigé

3)

```
# Suite de la page précédente
k = i+len('/CreationDate')
while texte[k] == ' ':
    k += 1
date = texte[k+1:k+1+23]
if date[:2] != 'D:':
    print("Fichier PDF. Format de date illisible")
    print(date)
    return
print("Fichier PDF créé le :")
print(date[8:10],'/',date[6:8],'/',date[2:6])
print(date[10:12],'h',date[12:14],'m',date[14:16],\
                                    's. GMT',date[16:19])
```

Exercice 3 - Enoncé

Exercice 3. *Un algorithme de recherche plus efficace*
Il existe des algorithmes de recherche de motif dans une chaine beaucoup plus efficaces. On peut procéder ainsi :
– Pour chercher si le motif recherché (de longueur `n`) apparaît à l'indice `i` on compare à partir de la fin du mot, c'est à dire si le caractère `mot[n-1]` est identique à `chaine[i+n-1]`,
– si c'est le cas on compare l'avant dernière lettre : `mot[n-2]` avec `chaine[i+n-2]`, etc... jusqu'à avoir trouvé le mot.
– si ce n'est pas le cas, on vérifie si le caractère `chaine[i+n-1]` est un caractère apparaissant dans `mot`. Dans le cas contraire on poursuit la recherche non pas à la position `i+1` mais à la position `i+n`.

Exercice 3 - Enoncé

1. Ecrire une fonction `caracteres` prenant en paramètre une chaine de caractère `mot` et qui renvoie une chaine contenant les mêmes caractères que `mot`, mais en seul exemplaire pour chacun.
2. Ecrire une fonction qui exécute cette recherche de motif par cette méthode.
3. Pouvez-vous en proposer une amélioration ? On ne demande pas de ma programmer.

Exercice 3 - Corrigé

1.

```
def caracteres(mot):
    ch = ""
    for x in mot:
    if x not in ch:
        ch += x
    return ch
```

Exercice 3 - Corrigé

2.

```
def recherche(chaine,mot):
    N = len(chaine)
    n = len(mot)
    carac = caracteres(mot)
    i = 0
    while i < N-n+1:
        k = n-1
        while k>=0 and chaine[i+k]==mot[k]:
            k-=1
        if k==-1:
            return True
        elif chaine[i+k] not in carac:
            i += n
        else:
            i += 1
    return False
```

Exercice 3 - Corrigé

3. Pour améliorer l'algorithme :

```
carac = caracteres(mot)
```

permet d'obtenir (prétraitement) la chaine des différents caractères de `mot`. On constitue aussi la liste des occurrences où ces caractères apparaissent dans `mot` :

```
carac, occur = occurence(mot)
```

en réécrivant la fonction `occurence` pour quel renvoie la chaine des caractères, et la liste de leurs occurrences. Par exemple :

```
carac, occur = occurence('abracadabra')
```

renverra : `carac : "abrcd"`
`occur : [[0,3,5,7,10], [1, 8], [2, 9], [4], [6]]`

Puis on modifie la fontion recherche : dans la partie `else:` plutôt que d'incrémenter `i` de 1, on fait varier `i` de façon à ne tester que les seules positions où le caractère `chaine[i+k]` apparaît dans `mot`, à l'aide de la liste `occur`.
On obtient un algorithme très efficace (Algorithme de Boyer-Moore).