Compte rendu du projet de fin de semestre

Recherche de mots dans un texte

LIN Gerald
SINNAPPAN Alain

Année 2016-2017

Licence math informatique
TDe - TP10

Sommaire :

I / Introduction :

Lors de l'écriture d'un texte ou de l'étude de celui-ci, On cherche parfois à retrouver un mot en particulier dans le texte que ce soit dans le but de centrer une information ou de trouver le lien qu'a le mot avec le texte. De nos jours, la plupart des éditeurs de textes proposent cette fonction et ainsi on a décidé de créer un logiciel qui aurait la même fonction mais à notre manière et seulement avec du python3.

On a choisi ce projet car il nous semblait plus intéressant de voir comment fonctionne cet outil et aussi parce qu'on voulait se démarquer des autres groupes qui, le plus souvent, choisissent un jeu comme projet.

L'outil de recherche qu'on a conçu, permet de rechercher un mot précis dans un texte mais aussi plusieurs mots a des lettres près que l'on peut choisir, des classe de mots précis et approchées.

II / Réalisation du projet

1) Après avoir choisi le projet, la question était de savoir comment on allait le réaliser et pour cela on a noté sur une feuille toutes les idées qui nous passaient par la tête et on a regardé nos cours pour savoir ce qui pouvait être réutilisé afin de réaliser le projet.

2) Maintenant qu'on savait comment il fallait s'y prendre pour se rapprocher de la réalisation du projet, on a commencé à réaliser le projet.

Ordre de réalisation du programme :

a) Menu et choix
b) Programme, forme et choix
c) Développement des choix
d) Fonction recherche

1/ On a programmé les fonctions menu auquel l'utilisateur a accès et les différents choix proposés dans celui-ci. Afin de mettre en place ces fonctions on a écrit le corps principal du programme qui indique ce qu'il se passe à chaque choix effectué.

2/ Pour chaque choix, on a indiqué dans le corps principal ce que le programme devait demander et renvoyer à l'utilisateur.

3/ A chaque nouveau choix qu'on programmait dans le corps principal, on réfléchissait aux fonctions éventuelles que nécessitaient ce choix, et une fois que l'idée était claire on écrivait les fonctions nécessaire et on les réutilisait dans la fonction principale.

Voici la liste des fonctions principales qui ont été nécessaire au bon fonctionnement du programme :

```
def menu():
    print("\nSaisie d'un nouveau texte
    print("Recherche d'une classe de mots
def choice():
    choix=int(input("Entrez votre choix (0-5) : "))
    while choix<0 or choix>5:
        choix=int(input("Entrez votre choix (0-5) : "))
    return choix
```

1/ Permet la mise en page du menu sur le terminal.

```
def supp_ponctuation(mot):
    lst=[",",".","!","?",";",":"]
    i=0
    new_mot=""
    for i in mot:          #parcourir
        if i not in lst:   #si le car
            new_mot+=i
    return new_mot
```

2/ On fait une liste dans laquelle on indique les différentes ponctuations, si le mot entré par l'utilisateur possède une de ces ponctuations, on les supprime.

```
def mot_exact(mot,texte):
    pos=1
    lst=texte.split()
    lst_pos=[]
    for i in lst:
        element=supp_ponctuation(i)
        if element==mot:
            lst_pos.append(str(pos))
        pos+=len(i)+1
    return lst_pos
```

3/ CHOIX 1 → On crée une variable pos= 1, puis on sépare le texte en liste. On parcourt la liste, en supprimant les ponctuations et si il y a correspondance entre l'élément et le mot demandé, alors on ajoute la position du mot. Pour connaître la position de chaque mot, on ajoute la longueur du mot précédent, plus un (correspond à l'espace) pour passer au mot suivant.

```python
def mot_approche(mot,k,texte):
    pos=1
    lst=texte.split()
    lst_pos=[[]]
    for word in lst:
        element=supp_ponctuation(word)
        if len(element)==len(mot):
            compteur_lettre=0
            i=0
            while i<len(mot):
                if element[i]!=mot[i]:
                    compteur_lettre+=1
                i+=1
            if compteur_lettre<=k:
                if compteur_lettre==0:
                    lst_pos[0].append(str(pos))
                else:
                    lst_pos.append([element,str(pos),compteur_lettre])
        pos+=len(word)+1
    return lst_pos
```

4/ CHOIX 2 → On réutilise le même principe que la fonction précédente au début sauf qu'ici si la longueur de l'élément est égale à celle du mot recherché alors on créé un compteur de lettre et un indice égal à 0. Ensuite, tant que le compteur est inférieur à la longueur du mot, si l'élément est différent du mot alors on incrémente le compteur de lettre. Si le compteur de lettre est toujours égal à 0, alors on ajoute la position du mot puis on passe au mot suivant.

```python
def classe_menu():
    print("Choisissez :")
    print('\n1 : "x" -> un caractère de', "l'alphabet considéré")
    print("\n2 : * -> un caractère quelconque")
    print("\n3 : [caractères] -> une classe de caractères, tels les intervalles => [a,e,i,o,u] ou [A..Z]")
    print("\n4 : #C -> caractère quelconque sauf un caractère C ou d'une classe de caractères C")
    print("\n5 : arrêter")

def classe_choix():
    choix=int(input("\nEntrez votre choix ( (1-5) / (5) pour arrêter) : "))
    while choix<1 or choix>5:
        choix=int(input("Entrez votre choix ( (1-5) / (5) pour arrêter) : "))
    return choix
```

5/ Menu pour éviter a l'utilisateur d'avoir à entrer les caractères au clavier (même principe que le 1er menu)

```
def classe_mot():
    classe_menu()
    lst_car=[]          #liste de la classe du mot
    choix_car=0
    p=""                #pour le repérage
    while choix_car!=5:
        choix_car=classe_choix()
```

6/ Pré-choix 3 et 4 → fonction qui est la base du choix 3 et 4 : cette fonction permet à l'utilisateur de choisir entre plusieurs types de caractère afin de créer sa classe de mots. Pour ce faire, les différents types de caractères choisis par l'utilisateur seront enregistrés dans une liste, qui elle même sera renvoyée à la fin du programme, grâce à laquelle on pourra vérifier si chaque chaque lettre des mots du texte correspond aux critères.

```
    if choix_car==1:
        lettre=input("Choisissez une lettre : ")
        while len(lettre)>1:                #sécurité
            lettre=input("Choisissez une lettre : ")
        lst_car.append(lettre)
        #repérage: ###
        p+=lettre+" "
        print("Actuellement :",p)
```

7/ ***(suite de la fonction précédente)*** 1er type de caractère qui demande une lettre spécifique : une variable « lettre » est initialisée et permettant à l'utilisateur d'écrire le caractère qu'il veut. Une sécurité est mise en place pour éviter que l'utilisateur entre plus de 2 caractères en même temps.

```
if choix_car==2:
    lst_car.append("*")
    #repérage: ###
    p+="* "
    print("Actuellement :",p)
```

8/ 2ème type de caractère qui demande une lettre quelconque : on ajoute simplement le caractère « * » à la liste (de la classe du mot).

```
if choix_car==3:
    cl=int(input("Tapez 1 : [a,e,i,o,u] \nTapez 2 : [A..Z] \nVotre choix : "))          #varial
    while cl<1 or cl>2:           #sécurité
        cl=int(input("Tapez 1 : [a,e,i,o,u] \nTapez 2 : [A..Z] \nVotre choix : "))

    #ex : [a,e,i,o,u]
    if cl==1:
        caractere=input('Entrez vos caractères une à une ("stop" pour arrêter) : ')
        intervalle=[]               #liste de l'intervalle des lettres qui sera ajoutée à la

        while caractere!="stop":
            if caractere in intervalle:             #sécurité pour empêcher une lettre en doub
                print("Ce caractère est déjà enregistré !\n")
            elif (len(caractere)!=1) or (caractere in ["","  "]) or not str.isalpha(caractere):
                print("Réessayez.")
            else:
                intervalle.append(caractere)
            caractere=input('Entrez vos caractères une à une ("stop" pour arrêter) : ')
        if len(intervalle)>0:             #sécurité pour ne pas avoir de sous-liste vide dans
            lst_car.append(intervalle)
```

9/ 3ème type de caractère : suite de lettres précises → une variable « caractère » est initialisée pour permettre à l'utilisateur d'entrer la lettre souhaitée, le programme entre dans une boucle, et tant que l'utilisateur n'entre pas « stop », il continuera à entrer des lettres. Il y a des sécurités pour éviter d'entrer deux fois une même lettre, une sécurité pour ne pas avoir deux lettres en même temps ou si l'utilisateur entre un espace.

```
#ex : [A..Z]
if cl==2:
    intervalle=[]
    premier=input("Première lettre de l'intervalle : ")
    while not str.isalpha(premier) or len(premier)!=1:
        premier=input("Première lettre de l'intervalle : ")
    deuxieme=input("Deuxième lettre de l'intervalle : ")
    while not str.isalpha(deuxieme) or len(deuxieme)!=1:
        deuxieme=input("Deuxième lettre de l'intervalle : ")

    while ord(deuxieme)<ord(premier):
        premier=input("Première lettre de l'intervalle : ")
        while not str.isalpha(premier) or len(premier)!=1:
            premier=input("Première lettre de l'intervalle : ")
        deuxieme=input("Deuxième lettre de l'intervalle : ")
        while not str.isalpha(deuxieme) or len(deuxieme)!=1:
            deuxieme=input("Deuxième lettre de l'intervalle : ")

    for numero in range(ord(premier),ord(deuxieme)+1):
        intervalle.append(chr(numero))
    lst_car.append(intervalle)
```

10/ l'intervalle entre des lettres → le programme demande un intervalle de lettres grâce aux codes ASCII. L'utilisateur entre les bornes, ainsi le programme regarde le code ASCII des lettres et parcourt l'intervalle entre les deux lettres en ajoutant chaque lettre à une liste « intervalle » initialisée plus tôt.

```
    lettre=input("Choisissez
lettre_tempo="#"+lettre
lst_car.append(lettre_tempo)
```

```
intervalle=["#"]
```

Résultat : `[['a', 'b'], '#a', ['#', 'b', 'c']`

11/ 4ème type de caractère : reprend le même principe que le 3eme sauf qu'on rajoute un « # ».

```
def classe_recherche(lst_caractere,texte):
    pos=1
    lst_txt=texte.split()
    lst_pos=[]
    if lst_caractere==['']:      #lorsqu'il n'y a aucune saisie de l'user
        return lst_pos
    for word in lst_txt:
        element=supp_ponctuation(word)
        if len(element)==len(lst_caractere)-1:      # "-1" car il y a la variable
            i=0
            compteur_erreur=0
            while i<len(lst_caractere)-1:           #je parcours les caractères s
                if len(lst_caractere[i])==1 and str.isalpha(lst_caractere[i][0]):
                    if element[i] not in lst_caractere[i]:                      #s
                        compteur_erreur+=1
                elif len(lst_caractere[i])>1 and lst_caractere[i][0]!="#":
                    if element[i] not in lst_caractere[i]:
                        compteur_erreur+=1
                elif len(lst_caractere[i])>1 and lst_caractere[i][0]=="#":
                    if element[i] in lst_caractere[i]:
                        compteur_erreur+=1
                i+=1
            if compteur_erreur==0:          #si le compteur d'erreurs est différe
                lst_pos.append(str(pos))
        pos+=len(word)+1
    return lst_pos
```

12/ CHOIX 3 → premièrement, la fonction vérifie la longueur du mot, si elle correspond à la longueur de la classe de mots. Ensuite, il suffit de vérifier si chaque lettre du mot correspond aux types de caractères sélectionnés par l'utilisateur.

```
if compteur_erreur<=k:
    if compteur_erreur==0:
        lst_pos[0].append(str(pos))
    else:
        lst_pos.append([element,str(pos),compteur_erreur])
```

13/ CHOIX 4 → la fonction commence de la même façon, sauf le nombre de lettres différentes, il y a une vérification, si le compteur de lettres différentes est inférieur au nombre d'erreurs souhaité.

14/ Le programme : il consiste tout simplement à afficher les éléments dans les listes renvoyées par les fonction.

III/ Problèmes et résolutions

-Au début du projet, nous avons presque perdu une semaine à cause d'une mal compréhension des consignes du projet. Pour remédier à ce problème, nous avons posé des questions au responsable de notre TP
-Des recherches sans succès lors de l'écriture de certaines fonctions. Pour y remédier, nous avons beaucoup cherché sur internet jusqu'à trouver la solution.