TP N°14 : Les fichiers corrigé

1 Lecture de fichier

Exercice 1

1. Que renvoie la méthode `read()` lorsque la fin du fichier est atteinte ?
2. Décrire le fonctionnement des commandes `f.readline()`, `readline(n)` et `readlines()` ?
3. Écrivez une fonction `lecture` qui lit le fichier « test.txt » ligne par ligne et affiche son contenu.
4. Chaque ligne de texte se termine par la séquence de caractères `\n` qui s'appelle *séquence d'échappement* et désigne un passage à la ligne. Adapter la fonction `lecture` pour ne pas afficher ces séquences `\n`.
5. Modifiez la fonction `lecture` pour afficher le contenu d'un fichier dont le nom est donné en paramètre. Validez cette nouvelle fonction avec le fichier « test.txt ».
6. Essayez cette fonction avec un autre fichier texte.
7. Essayez cette fonction avec un nom de fichier qui n'existe pas. Modifiez la fonction `lecture` de manière à afficher "Le fichier <nom du fichier> n'existe pas".

```python
# Q2
def lecture1():
    f = open("test.txt","r")
    line = f.readline()
    while line:
        print(line) #,end=""
        line = f.readline()

# tester la fonction
lecture1()

#Q3
def lecture2():
    f = open("test.txt","r")
    line = f.readline()
    while line:
        print(line.replace("\n",""))
        line = f.readline()

# tester la fonction
lecture2()

#Q4
def lecture3(file):
    f = open(file,"r")
    line = f.readline()
    while line:
        print(line.replace("\n",""))
        line = f.readline()

# tester la fonction
file = "test.txt"
lecture3(file)

#Q5
# tester la fonction avec nom de fichier inexistant
file = "test_2.txt"
lecture3(file)

#Q6
def lecture4(file):
    try:
```

```
        f = open(file,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file))
        return
    line = f.readline()
    while line:
        print(line.replace("\n",""))
        line = f.readline()

# tester la fonction avec nom de fichier inexistant
file = "test_2.txt"
lecture(file)
```

2 Écriture de fichier

Exercice 2

1. Écrivez une fonction `ecriture` qui écrit dans un fichier ce qui est saisi au clavier jusqu'à ce qu'une ligne vide soit saisie.
2. Testez votre fonction et observez en particulier que toutes les lignes saisies sont mises bout à bout dans le fichier.
3. Améliorer votre fonction `ecriture` de manière à ce que chaque ligne saisie soit sur une ligne dans le fichier écrit.
4. Écrivez une fonction `ajout` qui Ajoute quelques lignes dans un fichier (vérifiez que cela est bien réalisé).

```
#Q1
def ecriture():
    f = open('test_2.txt','w')
    while True:
        ch = input("tapez un text : ")
        if len(ch) > 0:
            f.write(ch)
        else :
            f.close() #enregistrer
            return

#Q2
# tester la fonction
ecriture()


#Q3
def ecriture():
    f = open('test_2.txt','w')
    while True:
        ch = input("tapez un text : ")
        if len(ch) > 0:
            f.write(ch + "\n")
        else :
            f.close() #enregistrer
            return

# tester la fonction
ecriture()

#Q4
def ajout():
    f = open('test_2.txt','a')
    while True:
        ch = input("Ajouter un text : ")
```

```
        if len(ch) > 0:
            f.write(ch + "\n")
        else :
            f.close() #enregistrer
            return

# tester la fonction
ajout()
```

Exercice 3

1. Écrivez une fonction `copier(fichier)` qui recopie un fichier texte. Le programme demandera à l'utilisateur le nom (avec chemin d'accès) de fichier et le recopier dans le même emplacement.
2. Écrire une fonction `comparer` qui compare les contenus de deux fichiers et signale la première différence rencontrée.
3. A partir de deux fichiers préexistants A et B, construire un fichier C qui contienne alternativement un élément de A, un élément de B, un élément de A, et ainsi de suite, jusqu'à atteindre la fin de l'un des deux fichiers originaux. Compléter ensuite C avec les éléments restants sur l'autre.

```
#Q1
def copier(file):
    try:
        f = open(file,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file))
        return
    file_name = file[:file.index('.')]
    extension = file[file.index('.'):]
    new_file_name = file_name + " (copie)" + extension
    f2 = open(new_file_name,"w")
    f2.write(f.read())
    f2.close()

# tester la fonction
copier("test.txt")

#Q2
def comparer(file1, file2):
    '''
    compare les contenus de deux fichiers et signale la première différence
    rencontrée.
    '''
    # ouvrir le premier fichier
    try:
        f1 = open(file1,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file1))
        return

    # ouvrir le deuxième fichier
    try:
        f2 = open(file2,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file2))
        return

    content1 = f1.read()
    content2 = f2.read()

    i = 0
    while i < len(content1) and i < len(content2) and content1[i] == content2[i]:
```

```
        i += 1

    if i == len(content1) == len(content2) :
        print ("aucune différence")
    elif i == len(content1) :
        print("Première différence dans {} à la position : {}".format(file2,i))
    elif i == len(content2) :
        print("Première différence dans {} à la position : {}".format(file1,i))
    else:
        print("Première différence à la position : {}".format(i))

# test la fonction
file1 = "comparer1.txt"
file2 = "comparer2.txt"
comparer(file1, file2)

#Q3
def alterner(file1, file2):
    # ouvrir le premier fichier
    try:
        f1 = open(file1,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file1))
        return

    # ouvrir le deuxième fichier
    try:
        f2 = open(file2,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file2))
        return

    content1 = f1.readlines(); print(content1); print(content1[0][-1]=='\n')
    # ajouter un retour à la ligne s'il n'existe pas
    if len(content1) > 0 :
        if content1[-1][-1]!='\n':
            content1[-1]+='\n'

    content2 = f2.readlines(); print(content2)

    # ajouter un retour à la ligne s'il n'existe pas
    if len(content2) > 0 :
        if content2[-1][-1]!='\n':
            content2[-1]+='\n'

    content3 = []

    i = 0
    while i < len(content1) and i < len(content2):
        content3 += [content1[i]]
        content3 += [content2[i]]
        i += 1


    if i == len(content1) and i < len(content2) :
        content3 += content2[i:]
    elif i == len(content2) and i < len(content1) :
        content3 += content1[i:]

    f3 = open("C.txt","w")
    f3.writelines(content3)
```

```
    f3.close()


# test la fonction
file1 = "A.txt"
file2 = "B.txt"
alterner(file1, file2)
f3 = open("C.txt","r")
print(f3.read())
```

Exercice 4

Écrire un script python qui permet de définir et d'appeler les fonctions suivantes :

1. Créer une fonction `saisie_listes(n,m)` qui saisie et retourne n listes chacune de m entiers positifs. Tester cette fonction pour n=10 et m=10.
2. Créer une fonction `remplir(fichier, listes)` qui enregistre les listes dans un fichier texte dont le chemin est passé en paramètres. Tester cette fonction pour enregistrer les listes crées dans la question 1 dans un fichier nommé «fichier1.txt».
   Exemple :
   La liste `[[12, 5], [23, 4], [25, 1]]` sera enregistrée dans le fichier text sous le format suivant :
   12,5
   23,4
   25,1
3. Créer une fonction `trier_listes_fichier(fichier_source, fichier_distination)` qui permet de lire les listes d'entiers à partir de `fichier_source`, trier les entiers de chaque liste dans l'ordre croissant et enregistrer les résultats dans le `fichier_distination`. Tester cette fonction avec `fichier_source` : «fichier1.txt» et `fichier_distination` : «fichier2.txt».
4. Créer une fonction `trier_lignes_fichier(fichier_source, fichier_distination)` qui permet de lire les listes d'entiers à partir de `fichier_source`, trier les listes d'entiers dans l'ordre croissant selon le premier entier de chaque liste et enregistrer les résultats dans le `fichier_distination`. Tester cette fonction avec `fichier_source` : «fichier2.txt» et `fichier_distination` : «fichier3.txt».
5. Créer une fonction `recherche(n,fichier)` qui retourne les numéros de lignes et numéros de colonnes de chaque occurrence de l'entier n dans le fichier passé en paramètres ou bien elle retourne `None` si l'entier n'existe pas. Tester cette fonction sur le fichier «fichier3.txt» pour un entier n saisi au clavier.

```
#Q1
def saisie_listes(n,m) :
    """ saisie et retourne n listes chacune de m entiers positifs."""
    L = []
    for i in range(n):
        liste = []
        for j in range(m):
            while True:
                try:
                    entier = int(input("Tapez un entier positif : "))
                    if entier > 0 :
                        liste.append(entier)
                        break
                    else:
                        print("Erreur : Entier négatif, retapez ...")
                except:
                    print("Erreur de saisie ! Essayez de nouveau.")
        L.append(liste)
    return L

# Tester cette fonction pour n=10 et m=10.
L = saisie_listes(3,2)
print(L)
```

```python
#Q2
def remplir(fichier, listes) :
    ''' enregistre les listes dans un fichier texte dont le chemin
        est passé en paramètres. '''

    f = open(fichier, "w")

    for i in range(len(listes)):
        liste = listes[i]
        liste = [ str(entier) for entier in liste]
        f.write(",".join(liste)+"\n")

    f.close()

# Tester cette fonction pour enregistrer les listes crées dans la question 1 dans
# un fichier nommé «fichier1.txt».
L = [[12, 5], [23, 4], [25, 1]]
import os
print(os.getcwd())
os.chdir("D:/python/TP_14_fichiers")
print(os.getcwd())
remplir("fichier1.txt", L)

#Q3
def lire_entiers(fichier_source):
    ''' lire les listes d'entiers à partir de fichier_source'''
    try:
        f = open(fichier_source,'r')
        line = f.readline()
        listes = []
        while line:
            liste = line.split(',')
            liste = [int(elt) for elt in liste]
            listes.append(liste)
            line = f.readline()
    except:
        print("Erreur :fichier {} n'existe pas".format(fichier_source))
        raise Exception

    return listes

def trier(listes):
    '''trier les entiers de chaque liste dans l'ordre croissant'''
    for liste in listes:
        liste.sort()

def trier_listes_fichier(fichier_source, fichier_distination) :
    '''
    - lire les listes d'entiers à partir de fichier_source,
    - trier les entiers de chaque liste dans l'ordre croissant
    - et enregistrer les résultats dans le fichier_distination.
    '''

    # lire les listes d'entiers à partir de fichier_source
    try:
        listes = lire_entiers(fichier_source)
        print(listes)
    except :
        return

    # trier les entiers de chaque liste dans l'ordre croissant
```

```
    trier(listes)

    # enregistrer les résultats dans le fichier_distination
    remplir(fichier_distination, listes)


# Tester cette fonction avec
# fichier_source : «fichier1.txt» et fichier_distination : «fichier2.txt»
trier_listes_fichier("fichier1.txt", "fichier2.txt")

# Q4
def trier_lignes_fichier(fichier_source, fichier_distination) :
    '''
    - lire les listes d'entiers à partir de fichier_source,
    - trier les listes d'entiers dans l'ordre croissant selon le premier
        entier de chaque liste
    - et enregistrer les résultats dans le fichier_distination.
    '''
    # lire les listes d'entiers à partir de fichier_source
    try:
        listes = lire_entiers(fichier_source)
        print(listes)
    except :
        return

    # trier les listes d'entiers dans l'ordre croissant selon le premier
    # entier de chaque liste
    listes.sort(key=lambda liste : liste[0])


    # enregistrer les résultats dans le fichier_distination
    remplir(fichier_distination, listes)

# Tester cette fonction avec
# fichier_source : «fichier2.txt» et fichier_distination : «fichier3.txt».
trier_lignes_fichier("fichier2.txt", "fichier3.txt")

# Q5
def recherche(n,fichier) :
    '''
    retourne les numéros de lignes et numéros de colonnes de chaque occurrence
    de l'entier n dans le fichier passé en paramètres ou bien elle retourne None
    si l'entier n'existe pas.
    '''
    try:
        f = open(fichier,'r')
        lines = f.readlines()
        positions = []
        for i in range(len(lines)):
            line = lines[i]
            start = 0
            if str(n) in line:
                j = start + line.index(str(n))
                positions.append((i,j))
                start  += line.index(str(n))+len(str(n))
                if start < len(line):
                    line = line[start:]
        return positions if len(positions)>0 else None
    except:
        print("Erreur :fichier {} n'existe pas".format(fichier))
        raise Exception
```

```
148
149
150 # Tester cette fonction sur le fichier «fichier3.txt» pour un entier
151 # n saisi au clavier.
152
153 def saisir_entier():
154     while True:
155         try:
156             return (int(input("tapez un entier : ")))
157         except :
158             print("Erreur de saisie ! Tapez de nouveau.")
159
160 n = saisir_entier()
161 positions = recherche(n,"fichier3.txt")
162 print(positions)
```

Exercice 5

1. Écrivez un script qui cherche un mot dans un fichier texte et affiche, pour chaque occurrence trouvée, le numéro de la ligne contenant ce mot et la position du mot dans cette ligne. Le programme demandera à l'utilisateur le nom (avec chemin d'accès) de fichier et le mot à chercher.
2. Modifier le script précédent pour qu'il puisse chercher un mot dans une liste de fichiers d'un répertoire donné. Le programme demandera le nom du répertoire (avec chemin d'accès), le mot à chercher et affichera dans ce cas : le nom du fichier, le numéro de ligne et la position de chaque occurrence du mot cherché.
3. Modifier le script précédent pour qu'il puisse chercher un mot dans une liste de fichiers d'un répertoire donné et récursivement dans les fichiers de ses sous-répertoires.

```
1  import sys
2
3  #Q1
4  def read_content(file):
5      try:
6          f = open(file,"r")
7      except FileNotFoundError:
8          print("Le fichier {} n'existe pas".format(file))
9          return
10     return f.readlines()
11
12 def occurrences(liste, word):
13     occ = []
14     for i in range(len(liste)):
15         line = liste[i]
16         start = 0
17         while word in line:
18             j = start + line.index(word)
19             occ.append((i,j))
20             start  += line.index(word)+len(word)
21             if start < len(line):
22                 line = line[start:]
23
24     return occ
25
26 if __name__ == '__main__':
27
28     try :
29         file = sys.argv[1] #argv[0] refers to the name of this script
30         word = sys.argv[2]
31         liste = read_content(file)
32         print(occurrences(liste,word))
33     except IndexError:
34         print("Ajouter un nom de fichier suivi du mot chercher")
```

```
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file))


#Q2

import sys
import os
import os.path as path



def read_content(file):
    try:
        f = open(file,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file))
        return
    return f.readlines()

def occurrences(liste, word):
    occ = []
    for i in range(len(liste)):
        line = liste[i]
        start = 0
        while word in line:
            j = start + line.index(word)
            occ.append((i,j))
            start  += line.index(word)+len(word)
            if start < len(line):
                line = line[start:]

    return occ

if __name__ == '__main__':

    try :
        directory = sys.argv[1] #argv[0] refers to the name of this script
        word = sys.argv[2]

        #recupérer liste des noms des fichiers
        L = os.listdir()
        #print("L=",L)

        #filter : garder seulement les fichiers (isfile)
        file_list=list(filter(lambda file_name : path.isfile(file_name), L))
        #filtrer : garder seulement les fichiers text
        file_list=list(filter(lambda file_name : file_name.count('.txt'),file_list))

        #
        all_occurrences = []

        for i in range(len(file_list)):
            file = file_list[i]
            lines_list = read_content(file)
            result = occurrences(lines_list,word)
            if result :
                all_occurrences += [(file, occurrences(lines_list,word))]

        print(all_occurrences)

```

```
    except IndexError:
        print("Ajouter un nom de dossier suivi du mot chercher")
    except :
        directory = "."
        word = ""


#Q3

import sys
import os
import os.path as path



def read_content(file):
    try:
        f = open(file,"r")
    except FileNotFoundError:
        print("Le fichier {} n'existe pas".format(file))
        return
    return f.readlines()

def occurrences(liste, word):
    occ = []
    for i in range(len(liste)):
        line = liste[i]
        start = 0
        while word in line:
            j = start + line.index(word)
            occ.append((i,j))
            start  += line.index(word)+len(word)
            if start < len(line):
                line = line[start:]

    return occ

def search_occurrences_by_directory(directory):
    os.chdir(directory)
    print("current working directory:",directory)

    #recupérer liste des noms des fichiers
    L = os.listdir()
    #print("L=",L)

    #filter files  : garder seulement les fichiers (isfile)
    file_list=list(filter(lambda file_name : path.isfile(file_name), L))

    #filter text files : garder seulement les fichiers text
    file_list=list(filter(lambda file_name : file_name.count('.txt'),file_list))
    print("file_list:",file_list)

    #filter directories
    dir_list=list(filter(lambda file_name : path.isdir(file_name), L))
    print("dir_list:",dir_list)

    #search in the current directory
    for i in range(len(file_list)):
        file = file_list[i]
        print("current file:",file)

```

```
157         lines_list = read_content(file)
158         result = occurrences(lines_list,word)
159         print("result:",result)
160         print("len(result):",len(result))
161 
162         if len(result)>0 :
163             all_occurrences.append([(directory + "\\"+ file, occurrences(lines_list,word))])
164             print("x")
165             print("all_occurrences:",all_occurrences)
166 
167     #repeat the search in every sub folder recursively
168     for dir_name in dir_list:
169         sub_dir = directory + "\\"+ dir_name
170         #print(directory + "\\"+ dir_name)
171         return search_occurrences_by_directory(sub_dir)
172 
173 if __name__ == '__main__':
174     global all_occurrences
175     all_occurrences = []
176 
177     try :
178         root_directory = sys.argv[1] #argv[0] refers to the name of this script
179         word = sys.argv[2]
180 
181         print("all_occurrences:",all_occurrences)
182         search_occurrences_by_directory(root_directory)
183 
184         print(all_occurrences)
185 
186     except IndexError:
187         print("Ajouter un nom de dossier suivi du mot chercher")
188     except :
189         directory = "."
190         word = ""
```