

## 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".

```

1  # Q2
2  def lecture1():
3      f = open("test.txt", "r")
4      line = f.readline()
5      while line:
6          print(line) #, end=""
7          line = f.readline()
8
9  # tester la fonction
10 lecture1()
11
12 #Q3
13 def lecture2():
14     f = open("test.txt", "r")
15     line = f.readline()
16     while line:
17         print(line.replace("\n", ""))
18         line = f.readline()
19
20 # tester la fonction
21 lecture2()
22
23 #Q4
24 def lecture3(file):
25     f = open(file, "r")
26     line = f.readline()
27     while line:
28         print(line.replace("\n", ""))
29         line = f.readline()
30
31 # tester la fonction
32 file = "test.txt"
33 lecture3(file)
34
35 #Q5
36 # tester la fonction avec nom de fichier inexistant
37 file = "test_2.txt"
38 lecture3(file)
39
40 #Q6
41 def lecture4(file):
42     try:

```

```

43     f = open(file,"r")
44     except FileNotFoundError:
45         print("Le fichier {} n'existe pas".format(file))
46         return
47     line = f.readline()
48     while line:
49         print(line.replace("\n",""))
50         line = f.readline()
51
52     # tester la fonction avec nom de fichier inexistant
53     file = "test_2.txt"
54     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).

```

1     #Q1
2     def ecriture():
3         f = open('test_2.txt','w')
4         while True:
5             ch = input("tapez un text : ")
6             if len(ch) > 0:
7                 f.write(ch)
8             else :
9                 f.close() #enregistrer
10                return
11
12     #Q2
13     # tester la fonction
14     ecriture()
15
16
17     #Q3
18     def ecriture():
19         f = open('test_2.txt','w')
20         while True:
21             ch = input("tapez un text : ")
22             if len(ch) > 0:
23                 f.write(ch + "\n")
24             else :
25                 f.close() #enregistrer
26                return
27
28     # tester la fonction
29     ecriture()
30
31     #Q4
32     def ajout():
33         f = open('test_2.txt','a')
34         while True:
35             ch = input("Ajouter un text : ")

```

```

36         if len(ch) > 0:
37             f.write(ch + "\n")
38         else :
39             f.close() #enregistrer
40             return
41
42 # tester la fonction
43 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.

```

1  #Q1
2  def copier(file):
3      try:
4          f = open(file,"r")
5      except FileNotFoundError:
6          print("Le fichier {} n'existe pas".format(file))
7          return
8      file_name = file[:file.index('.')]
9      extension = file[file.index('.'):]
10     new_file_name = file_name + " (copie)" + extension
11     f2 = open(new_file_name,"w")
12     f2.write(f.read())
13     f2.close()
14
15 # tester la fonction
16 copier("test.txt")
17
18 #Q2
19 def comparer(file1, file2):
20     '''
21     compare les contenus de deux fichiers et signale la première différence
22     rencontrée.
23     '''
24     # ouvrir le premier fichier
25     try:
26         f1 = open(file1,"r")
27     except FileNotFoundError:
28         print("Le fichier {} n'existe pas".format(file1))
29         return
30
31     # ouvrir le deuxième fichier
32     try:
33         f2 = open(file2,"r")
34     except FileNotFoundError:
35         print("Le fichier {} n'existe pas".format(file2))
36         return
37
38     content1 = f1.read()
39     content2 = f2.read()
40
41     i = 0
42     while i < len(content1) and i < len(content2) and content1[i] == content2[i]:

```

```

43     i += 1
44
45     if i == len(content1) == len(content2) :
46         print ("aucune différence")
47     elif i == len(content1) :
48         print("Première différence dans {} à la position : {}".format(file2,i))
49     elif i == len(content2) :
50         print("Première différence dans {} à la position : {}".format(file1,i))
51     else:
52         print("Première différence à la position : {}".format(i))
53
54     # test la fonction
55     file1 = "comparer1.txt"
56     file2 = "comparer2.txt"
57     comparer(file1, file2)
58
59     #Q3
60     def alterner(file1, file2):
61         # ouvrir le premier fichier
62         try:
63             f1 = open(file1,"r")
64         except FileNotFoundError:
65             print("Le fichier {} n'existe pas".format(file1))
66             return
67
68         # ouvrir le deuxième fichier
69         try:
70             f2 = open(file2,"r")
71         except FileNotFoundError:
72             print("Le fichier {} n'existe pas".format(file2))
73             return
74
75         content1 = f1.readlines(); print(content1); print(content1[0][-1]=='\n')
76         # ajouter un retour à la ligne s'il n'existe pas
77         if len(content1) > 0 :
78             if content1[-1][-1]!='\n':
79                 content1[-1]+='\n'
80
81         content2 = f2.readlines(); print(content2)
82
83         # ajouter un retour à la ligne s'il n'existe pas
84         if len(content2) > 0 :
85             if content2[-1][-1]!='\n':
86                 content2[-1]+='\n'
87
88         content3 = []
89
90         i = 0
91         while i < len(content1) and i < len(content2):
92             content3 += [content1[i]]
93             content3 += [content2[i]]
94             i += 1
95
96
97         if i == len(content1) and i < len(content2) :
98             content3 += content2[i:]
99         elif i == len(content2) and i < len(content1) :
100             content3 += content1[i:]
101
102         f3 = open("C.txt", "w")
103         f3.writelines(content3)

```

```

104     f3.close()
105
106
107     # test la fonction
108     file1 = "A.txt"
109     file2 = "B.txt"
110     alterner(file1, file2)
111     f3 = open("C.txt", "r")
112     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 créées 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.

```

1  #Q1
2  def saisie_listes(n,m) :
3      """ saisie et retourne n listes chacune de m entiers positifs. """
4      L = []
5      for i in range(n):
6          liste = []
7          for j in range(m):
8              while True:
9                  try:
10                     entier = int(input("Tapez un entier positif : "))
11                     if entier > 0 :
12                         liste.append(entier)
13                         break
14                     else:
15                         print("Erreur : Entier négatif, retapez ...")
16                 except:
17                     print("Erreur de saisie ! Essayez de nouveau.")
18             L.append(liste)
19     return L
20
21     # Tester cette fonction pour n=10 et m=10.
22     L = saisie_listes(3,2)
23     print(L)
24
25

```

```

26 #Q2
27 def remplir(fichier, listes) :
28     ''' enregistre les listes dans un fichier texte dont le chemin
29         est passé en paramètres. '''
30
31     f = open(fichier, "w")
32
33     for i in range(len(listes)):
34         liste = listes[i]
35         liste = [ str(entier) for entier in liste]
36         f.write(",".join(liste)+"\n")
37
38     f.close()
39
40 # Tester cette fonction pour enregistrer les listes créées dans la question 1 dans
41 # un fichier nommé «fichier1.txt».
42 L = [[12, 5], [23, 4], [25, 1]]
43 import os
44 print(os.getcwd())
45 os.chdir("D:/python/TP_14_fichiers")
46 print(os.getcwd())
47 remplir("fichier1.txt", L)
48
49 #Q3
50 def lire_entiers(fichier_source):
51     ''' lire les listes d'entiers à partir de fichier_source'''
52     try:
53         f = open(fichier_source, 'r')
54         line = f.readline()
55         listes = []
56         while line:
57             liste = line.split(',')
58             liste = [int(elt) for elt in liste]
59             listes.append(liste)
60             line = f.readline()
61     except:
62         print("Erreur :fichier {} n'existe pas".format(fichier_source))
63         raise Exception
64
65     return listes
66
67 def trier(listes):
68     '''trier les entiers de chaque liste dans l'ordre croissant'''
69     for liste in listes:
70         liste.sort()
71
72 def trier_listes_fichier(fichier_source, fichier_distination) :
73     '''
74     - lire les listes d'entiers à partir de fichier_source,
75     - trier les entiers de chaque liste dans l'ordre croissant
76     - et enregistrer les résultats dans le fichier_distination.
77     '''
78
79     # lire les listes d'entiers à partir de fichier_source
80     try:
81         listes = lire_entiers(fichier_source)
82         print(listes)
83     except :
84         return
85
86     # trier les entiers de chaque liste dans l'ordre croissant

```

```

87     trier(listes)
88
89     # enregistrer les résultats dans le fichier_destination
90     remplir(fichier_destination, listes)
91
92
93     # Tester cette fonction avec
94     # fichier_source : «fichier1.txt» et fichier_destination : «fichier2.txt»
95     trier_listes_fichier("fichier1.txt", "fichier2.txt")
96
97     # Q4
98     def trier_lignes_fichier(fichier_source, fichier_destination) :
99         '''
100         - lire les listes d'entiers à partir de fichier_source,
101         - trier les listes d'entiers dans l'ordre croissant selon le premier
102           entier de chaque liste
103         - et enregistrer les résultats dans le fichier_destination.
104         '''
105         # lire les listes d'entiers à partir de fichier_source
106         try:
107             listes = lire_entiers(fichier_source)
108             print(listes)
109         except :
110             return
111
112         # trier les listes d'entiers dans l'ordre croissant selon le premier
113         # entier de chaque liste
114         listes.sort(key=lambda liste : liste[0])
115
116
117         # enregistrer les résultats dans le fichier_destination
118         remplir(fichier_destination, listes)
119
120     # Tester cette fonction avec
121     # fichier_source : «fichier2.txt» et fichier_destination : «fichier3.txt».
122     trier_lignes_fichier("fichier2.txt", "fichier3.txt")
123
124     # Q5
125     def recherche(n,fichier) :
126         '''
127         retourne les numéros de lignes et numéros de colonnes de chaque occurrence
128         de l'entier n dans le fichier passé en paramètres ou bien elle retourne None
129         si l'entier n'existe pas.
130         '''
131         try:
132             f = open(fichier,'r')
133             lines = f.readlines()
134             positions = []
135             for i in range(len(lines)):
136                 line = lines[i]
137                 start = 0
138                 if str(n) in line:
139                     j = start + line.index(str(n))
140                     positions.append((i,j))
141                     start += line.index(str(n))+len(str(n))
142                 if start < len(line):
143                     line = line[start:]
144             return positions if len(positions)>0 else None
145         except:
146             print("Erreur :fichier {} n'existe pas".format(fichier))
147             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")

```

```

35     except FileNotFoundError:
36         print("Le fichier {} n'existe pas".format(file))
37
38
39 #Q2
40
41 import sys
42 import os
43 import os.path as path
44
45
46
47 def read_content(file):
48     try:
49         f = open(file,"r")
50     except FileNotFoundError:
51         print("Le fichier {} n'existe pas".format(file))
52         return
53     return f.readlines()
54
55 def occurrences(liste, word):
56     occ = []
57     for i in range(len(liste)):
58         line = liste[i]
59         start = 0
60         while word in line:
61             j = start + line.index(word)
62             occ.append((i,j))
63             start += line.index(word)+len(word)
64             if start < len(line):
65                 line = line[start:]
66
67     return occ
68
69 if __name__ == '__main__':
70
71     try :
72         directory = sys.argv[1] #argv[0] refers to the name of this script
73         word = sys.argv[2]
74
75         #recupérer liste des noms des fichiers
76         L = os.listdir()
77         #print("L=",L)
78
79         #filter : garder seulement les fichiers (isfile)
80         file_list=list(filter(lambda file_name : path.isfile(file_name), L))
81         #filtrer : garder seulement les fichiers text
82         file_list=list(filter(lambda file_name : file_name.count('.txt'),file_list))
83
84         #
85         all_occurrences = []
86
87         for i in range(len(file_list)):
88             file = file_list[i]
89             lines_list = read_content(file)
90             result = occurrences(lines_list,word)
91             if result :
92                 all_occurrences += [(file, occurrences(lines_list,word))]
93
94         print(all_occurrences)
95

```

```

96     except IndexError:
97         print("Ajouter un nom de dossier suivi du mot chercher")
98     except :
99         directory = "."
100        word = ""
101
102
103    #Q3
104
105    import sys
106    import os
107    import os.path as path
108
109
110
111    def read_content(file):
112        try:
113            f = open(file,"r")
114        except FileNotFoundError:
115            print("Le fichier {} n'existe pas".format(file))
116            return
117        return f.readlines()
118
119    def occurrences(liste, word):
120        occ = []
121        for i in range(len(liste)):
122            line = liste[i]
123            start = 0
124            while word in line:
125                j = start + line.index(word)
126                occ.append((i,j))
127                start += line.index(word)+len(word)
128                if start < len(line):
129                    line = line[start:]
130
131        return occ
132
133    def search_occurrences_by_directory(directory):
134        os.chdir(directory)
135        print("current working directory:",directory)
136
137        #recupérer liste des noms des fichiers
138        L = os.listdir()
139        #print("L=",L)
140
141        #filter files : garder seulement les fichiers (isfile)
142        file_list=list(filter(lambda file_name : path.isfile(file_name), L))
143
144        #filter text files : garder seulement les fichiers text
145        file_list=list(filter(lambda file_name : file_name.count('.txt'),file_list))
146        print("file_list:",file_list)
147
148        #filter directories
149        dir_list=list(filter(lambda file_name : path.isdir(file_name), L))
150        print("dir_list:",dir_list)
151
152        #search in the current directory
153        for i in range(len(file_list)):
154            file = file_list[i]
155            print("current file:",file)
156

```

```
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 = ""
```