Master 1 Humanités numériques – Algorithmique et programmation

TD 6 : Fichiers textes

J. Darmont – https://eric.univ-lyon2.fr/jdarmont/

L'objectif de ce TD est d'extraire le dictionnaire d'un fichier texte (l'ensemble des mots composant le texte). Les exercices sont à formuler en langage algorithmique, puis en Python. Ne pas oublier la déclaration des variables en algorithmique. Tester l'exécution avec Python.

Exercice 1 : Lecture d'un fichier

1. Télécharger le fichier lorem-ipsum.txt[1] et le placer dans l'espace de stockage du *notebook* (cf. TD 4).
2. En algorithmique, on suppose que la procédure *afficherListe* écrite au TD 5 est disponible. En Python, télécharger le fichier listeMod.py[2] créé au TD 5 et le placer dans l'espace de stockage du *notebook*. Dans une cellule de code, importer le module listeMod (cf. TD 4).
3. Ouvrir le fichier *f* correspondant à lorem-ipsum.txt en lecture.
4. Lire chaque ligne du fichier *f* et les placer dans une liste *listeLignes*.
5. Fermer le fichier *f*.
6. Afficher la liste *listeLignes*. Vous devez obtenir autant de lignes que le nombre de paragraphes dans le fichier (5).

Exercice 2 : Traitement du contenu du fichier

On souhaite extraire de la liste des lignes du fichier les mots qui les composent pour constituer un dictionnaire. Pour cela, ajouter à l'algorithme et au programme précédents les traitements suivants.

Pour chaque chaque ligne de *listeLignes* :

1. supprimer successivement les points, les virgules et les retours charriot (caractère spécial \n) de la ligne en les remplaçant par une chaîne vide ;
2. transformer la ligne en une liste de chaînes nommée *lesMots* ;
3. afficher la liste *lesMots* pour vérification. Tester. Vous devez obtenir autant de listes de mots que le nombre de lignes.

1 https://eric.univ-lyon2.fr/jdarmont/docs/lorem-ipsum.txt
2 https://eric.univ-lyon2.fr/jdarmont/docs/listeMod.py

4 remplacer l'affichage de *lesMots* par son ajout dans la liste de chaînes *dico*, qui aura dû être initialisée vide avant la boucle « pour chaque ligne ». L'union de listes se fait avec le symbole ∪ en algorithmique et avec le + en Python.

5 Après la boucle « pour chaque ligne », afficher la liste *dico*. Remarque ?

Afin de dédoublonner les mots du dictionnaire, nous allons utiliser des sous-programmes tout faits :

- Algorithmique : procédure Dédoublonner(uneListe) ;
- Python : fonction set(*uneListe*), qui transforme une liste en *ensemble* sans doublon. Pour retransformer l'ensemble en liste, il faut lui appliquer la fonction list(). Il est possible d'appliquer une fonction à une autre fonction, par exemple : *uneListe* = list(set(*uneListe*)).

6 Dédoublonner, puis trier la liste *dico*.

7 Afficher la liste *dico*.

Exercice 3 : Écriture d'un fichier

1 Toujours à la suite de l'algorithme et du programme précédents, ouvrir un nouveau fichier *f2* nommé dico.txt en écriture.

2 Écrire chaque mot du dictionnaire *dico* dans le fichier *f2*.

3 Fermer le fichier *f2*.

4 Consulter le fichier dico.txt pour vérifier que la liste des mots y est bien sauvegardée (double clic sur le fichier dans l'espace de stockage du *notebook*.

Questions subsidiaires (Python)

1 Transformer le programme développé dans les exercices 1 à 3 en une fonction nommée *extraireDictionnaire* prenant en paramètres le nom du fichier texte à traiter en entrée et le nom du fichier dictionnaire en sortie.

2 Placer la fonction *extraireDictionnaire* dans un module nommé dicoMod.

Correction

```
Algorithme lireEtTransformer
    Var f, f2 : Fichiers
    Var ligne : Chaîne
    Var listeLignes, dico, lesMots : Listes de Chaînes
Début
    {Initialisation de listeLignes}
    listeLignes ← ( )
    {Lecture et affichage du fichier d'entrée}
    f ← Ouvrir("lorem-ipsum.txt", "lecture")
    Lire(f, ligne)
    Tant que non FinDeFichier(f) faire
        listeLignes( ) ← ligne                  {Ajout dans listeLignes}
        Lire(f, ligne)
    Fin tant que
    Fermer(f)
    afficherListe(listeLignes)
    {Extraction des mots}
    dico ← ( )
    Pour ligne dans listeLignes faire
        ligne ← Remplacer(ligne, ".", "")       {Suppression des points}
        ligne ← Remplacer(ligne, ",", "")       {Suppression des virgules}
        ligne ← Remplacer(ligne, "\n", "")      {Suppression des retours chariot}
        lesMots ← Découper(ligne, " ")          {Extraction des mots}
        dico ← dico ∪ lesMots
    Fin pour
    afficherListe(dico)
    {Dédoublonnage et tri du dictionnaire}
    Dédoublonner(dico)
    Trier(dico)
    afficherListe(dico)
    {Écriture du dictionnaire dans un fichier de sortie}
    f2 ← Ouvrir("dico.txt", "écriture")
    Pour ligne dans dico faire
        Écrire(f2, ligne)
    Fin pour
    Fermer(f2)
Fin
```

```
# Importation de la fonction afficherListe
from listeMod import *

# Lecture et affichage du fichier d'entrée
f = open("lorem-ipsum.txt", "r")
listeLignes = f.readlines()
f.close()
afficherListe(listeLignes)

# Extraction des mots
dico = [ ]
for ligne in listeLignes:
	ligne = ligne.replace(".", "")	# Suppression des points
	ligne = ligne.replace(",", "")	# Suppression des virgules
	ligne = ligne.replace("\n", "")	# Suppression des retours chariot
	lesMots = ligne.split(" ")	# Extraction des mots
	dico = dico + lesMots
afficherListe(dico)

# Dédoublonnage et tri du dictionnaire
dico = list(set(dico))
dico.sort()
afficherListe(dico)

# Écriture du dictionnaire dans un fichier de sortie
f2 = open("dico.txt", "w")
for mot in dico:
	f2.write(mot + "\n")
f2.close()



# Questions subsidiaires
# Placer la fonction ci-dessous dans un fichier nommé dicoMod.py
def extraireDictionnaire(nomFichierEntree, nomDicoSortie):
	# Reproduire ci-dessous tout le code ci-dessus :)
	# Dans la section "Lecture et affichage du fichier d'entrée",
	# remplacer f = open("lorem-ipsum.txt", "r") par
	f = open(nomFichierEntree, "r")
	# Dans la section "Ecriture du dictionnaire dans un fichier de sortie",
	# remplacer f2 = open("dico.txt", "w") par
	f2 = open(nomDicoSortie, "w")

# Exemple d'utilisation
from dicoMod import extraireDictionnaire
extraireDictionnaire("lorem-ipsum.txt", "dico-lorem.txt")
```