Fichiers: généralités

1) Définition d'un fichier, nom et extension

Un fichier informatique est une suite ordonnée d'octets enregistrés sur un support de stockage à mémoire non volatile comme un disque dur, une clé USB ou un DVD. Certains logiciels, comme EditHexa, sont des éditeurs hexadécimaux permettant de voir ou de modifier n'importe quel octet d'un fichier quelconque.

Un fichier a un nom et souvent un suffixe, l'extension du fichier, qui renseigne l'OS (operating system ou encore système d'exploitation) sur la nature des informations contenues dans le fichier et le logiciel utilisé par défaut pour l'ouvrir, c'est-à-dire l'exécuter. Nom et extension sont séparés par un point. Le nom avec extension du fichier peut être par exemple script.bat , lettre.doc, notepad.exe, cigale.txt.

Dans l'explorateur de fichiers, l'extension est souvent masquée par défaut. Mais on peut changer cela et voir le nom complet des fichiers.

2) Fichiers programmes, fichiers de données

Il y a deux sortes de fichiers:

a) Les fichiers programmes

Les fichiers programmes, ou encore fichiers exécutables contiennent du code directement exécutable par le microprocesseur. Ils sont identifiés par le système d'exploitation par leur extension, .exe ou .bat par exemple dans Windows.

b) Les fichiers de données

Les fichiers de données contiennent du code qui n'est pas directement exécutable par le microprocesseur. Ils doivent être ouverts (exécutés) par un fichier programme. Leur extension précise la nature des données qu'ils contiennent, et permet aux fichiers programmes de "savoir" s'ils sont capables ou non d'ouvrir le fichier de données. Pour chaque extension de fichier de données il y a un programme par défaut (on peut le changer) destiné à l'ouvrir.

Par exemple, les fichiers avec l'extension .html vont être ouverts par le navigateur, les fichiers .odt par Libre Office Writer, les fichiers .py par Python, etc…Les fichiers textes (d'extension .txt) sont des fichiers de données et sont ouverts par Notepad.exe dans Windows, mais beaucoup d'autres logiciels peuvent ouvrir et travailler avec des fichiers textes.

3) Organisation des fichiers

En vue de faciliter leur localisation et donc de garantir un accès rapide, les fichiers sont classés par l'OS dans une structure arborescente par le système de gestion de fichiers.

Pour l'utilisateur, un système de fichiers est vu comme une arborescence : les fichiers sont regroupés dans des conteneurs appelés répertoires ou dossiers. Ces répertoires contiennent soit des fichiers, soit d'autres répertoires. Il y a donc un répertoire racine et des sous-répertoires. Une telle organisation génère une hiérarchie de répertoires et de fichiers organisés en arbre.

4) Chemin d'accès à un fichier ou à un répertoire

Le chemin d'accès à un fichier est une chaîne de caractères décrivant la place du fichier dans l'arborescence. Ce chemin peut être:

a) Un chemin absolu

La position du fichier est décrite depuis la racine de l'arbre, qui est sous Windows le nom de la partition (comme C: ou E:) dans laquelle se trouve le fichier. On sépare les noms des répertoires successifs depuis la racine de l'arbre jusqu'à la "branche" contenant le fichier (que l'on peut voir comme une feuille de l'arbre) par le caractère \ (antislash) dans Windows, et par un caractère / (slash) dans linux ou sur le web.

Par exemple si un fichier a pour chemin absolu C:\documents\lettres\modele.txt, cela signifie que son nom est modele.txt, qu'il est situé dans le répertoire lettres qui lui-même est situé dans le répertoire documents qui lui-même est à la racine du lecteur C:

b) Un chemin relatif

La position du fichier (ou du répertoire) est décrite depuis le répertoire où l'on est (appelé répertoire courant ou répertoire de travail), ce qui implique que le fichier soit dans l'un des sous répertoires.

En reprenant l'exemple précédent, depuis le répertoire C:\documents, notre fichier a pour chemin relatif lettres\modele.txt

Fichiers textes : lire, écrire, chercher

I) Travailler avec les fichiers textes en Python

1) Modules utiles, notation d'un chemin

Le module os contient les outils permettant d'interagir avec le système d'exploitation. Par exemple:

```
import os
os.getcwd()       # Renvoie le répertoire de travail ou encore répertoire courant
os.chdir(chemin)  # Change le répertoire courant, chemin est une chaîne de caractères
```

Dans Windows, les chemins des fichiers sont écrits avec des antislashs ou encore backslash (\). Mais le caractère \ est en Python le caractère d'échappement dans une chaîne de caractères. Et donc en Python le caractère "\" est "\\" . Il faudra donc pour ne pas avoir d'ennuis doubler le backslash dans les noms de chemins, qui peuvent être rentrés par des copies depuis la barre d'affichage de l'explorateur windows. Par exemple:

```
chemin = "C:\Users\Eric\Documents\Algorithmique"  # chemin windows devra être écrit
chemin = "C:\\Users\\Eric\\Documents\A\lgorithmique"  # comme chemin Python correct
```

On peut aussi écrire les chemins avec des slashs (/).

```
chemin = "C:/Users/Eric/Documents/Algorithmique"  # est correct
```

2) Ecrire ou lire dans un fichier texte

Python doit ouvrir le fichier, puis lire ou écrire et enfin fermer le fichier. L'utilisation facultative du mot clé "with" permet de travailler en lecture ou écriture un fichier sans avoir besoin de le fermer avec la méthode close(), et en libérant la ressource (le fichier) quelles que soient les éventuelles erreurs qui se pourraient se produire.

Python a trois modes d'ouverture d'un fichier texte:

(r) Lecture

(w) Ecriture en écrasant l'actuel fichier ou en le créant le fichier s'il n'existe pas

(a) Ecriture en ajoutant les données au fichier s'il existe déjà ou en le créant le fichier s'il n'existe pas

On peut préciser la norme d'encodage du fichier si on a des doutes ou si on veut la modifier.

Les instructions suivantes montrent comment ouvrir un fichier en lecture ou écriture et diverses possibilités pour lire ou écrire.

```
with open(nom_fichier,"r", encoding = "utf8") as fich:      # Ouverture en lecture
# contenu est une variable de type str dans laquelle il y a tout le contenu du fichier
   contenu = fich.read()
# liste_lignes est une variable de type list dont les éléments sont les lignes du fichier
   liste_lignes = fich.readlines()         # Liste des lignes du fichier
# pour travailler sur les lignes du fichier, on peut aussi faire comme ceci:
   for ligne in fich:
      faire quelque chose avecligne
```

```
with open(nom_fichier,"w", encoding = "utf8") as fich:      # Ouverture en écriture
   fich.write("texte")                    # Ecrire un texte sans aller à la ligne
   fich.write("texte\n")                  # Ecrire un texte et aller à la ligne
   fich.writelines(liste_textes)          # Ecrire une liste de textes
```

Le paramètre `encoding = "utf8"` est facultatif. S'il est omis, la norme d'encodage est la norme par défaut de l'OS.

3) Problèmes liés à l'encodage des fichiers textes

La norme Unicode attribue à chaque caractère ou symbole de toutes (c'est l'objectif) les langues du monde un identifiant unique: un entier, appelé point de code. Dans Python la fonction chr() permet d'obtenir le caractère correspondant à un code donné.

Par exemple, chr(8704) = ∀.

Un fichier informatique étant une suite finie d'octets, il faut définir pour chaque caractère du texte la suite finie d'octets le représentant dans le fichier. C'est la norme d'encodage qui précise cela. Pour des raisons historiques (entre autres minimiser la place en mémoire), il en existe beaucoup. Les plus utilisées sont:

- la norme historique ASCII
- la norme Latin-1, utilisée en Europe occidentale
- la norme ANSI ou Windows-1252, utilisée par l'OS Windows
- la norme UTF8, qui tend à s'imposer et qui permet d'encoder tous les caractères du jeu Unicode.

La norme mbcs est une norme propre à Python qui est adaptée à un environnement Windows.

Pour qu'un logiciel (par exemple Python) puisse lire correctement un fichier texte, il doit savoir dans quelle norme il a été encodé. Sans connaitre la norme d'encodage, il utilisera la norme par défaut de l'OS. Le plus souvant tout ira bien, mais pas toujours... De même, si on veut entregistrer un texte dans une norme différente de la norme par défaut de l'OS, il faudra la préciser.

II) Exercices

Q1) Affichier le répertoire de travail et (s'il le faut) changez ce répertoire courant de façon à ce qu'il soit un répertoire personnel du réseau du lycée, d'une clé usb ou de votre PC dédié à ce TP,

Q2) Récupérer (clic droit et "Enregistrer la cible du lien sous") le fichier "cigale.txt" contenant la fable "La cigale et la fourmi" sur le site internet maths-algo.fr et recopiez le dans votre répertoire de travail et **faites en une copie de sauvegarde**. Ce fichier cigale.txt est encodé avecla norme utf8.

Q3) Essais d'ouverture en lecture:

- Ecrire une fonction cigale_totale() qui ouvre le fichier "cigale.txt" et en affiche le contenu en totalité.
- Ecrire une fonction cigale_partielle() qui ouvre le fichier "cigale.txt" et en affiche une ligne sur deux.
- Ecrire une fonction cigale_point() qui ouvre le fichier "cigale.txt" et affiche les lignes se terminant par un point.

Q4) Essais d'ouverture en écriture:

- Ecrire une fonction cigale_morte() qui ouvre en mode (w) le fichier cigale.txt sans rien faire d'autre: qu'est devenu le contenu du fichier, est-ce normal ? Restaurez le fichier avec votre copie de sauvegarde.
- Ecrire une fonction cigale_idem() qui ouvre en mode (a) le fichier cigale.txt sans rien faire d'autre: qu'est devenu le contenu du fichier, est-ce normal ?
- Ecrire une fonction cigale_copie() qui ouvre en lecture "cigale.txt", en récupère le contenu, puis qui ouvre en écriture un nouveau fichier appelé "cigale_cop.txt" dans lequel on met le contenu du fichier "cigale.txt" (vous venez de aire une copie...)
- Ecrire une fonction cigale_auteur() qui crée un nouveau fichier appelé "cigale auteur.txt" dont la première ligne est "Jean de La Fontaine" suivi d'une ligne vide puis ensuite le contenu de "cigale.txt"
- Ecrire une fonction cigale_ordonnée() qui crée un nouveau fichier appelé "cigale ordonnée.txt" dont les lignes sont les lignes du fichier "cigale.txt" triées par ordre croissant.

Les premières lignes du fichier "cigale ordonnée" seront:

"Je vous paierai, lui dit-elle,
- Nuit et jour à tout venant
- Vous chantiez ? j'en suis fort aise.
Avant l'Oût, foi d'animal,

III) Recherche d'un mot dans un fichier

On veut savoir si un un mot est présent dans un fichier texte donné. Lorsque la réponse est positive, on souhaite connaître la première occurence et le nombre d'occurrences du mot dans le fichier. Python propose les fonctions count et find permettant de répondre à ces questions mais on s'interdit leur utilisation ici, sauf comme contrôle.

Les fichiers de données, cigale.txt et vie.txt (un roman de Guy de Maupassant) sont disponibles sur le site internet maths-algo.fr. Récupérer les fichiers de données et les recopier dans votre répertoire de travail.

On cherche à savoir si un mot donné est présent dans un fichier et où il se trouve pour la première fois.

Q5) Ecrire une fonction premiere_occurrence(mot, texte) (avec mot et texte de type str) qui renvoie:

- -1 si mot ne se trouve pas dans texte
- le plus petit entier k tel que mot se trouve pour la première fois en position k dans texte.

On proposera deux versions:

- une version n'utilisant pas les slices, qui ne fait que des comparaisons de caractères
- •• une version utilisant les "slices" (les tranches de liste) et faisant des comparaisons de listes. On rappelle que si s est une liste, s[a : b] est la tranche de liste [s[a], s[a+1],...,s[b-1]]

Noter: `nM, nT = len(mot), len(texte)` . Vérifier par exemple que :

```
fich = contenu du fichier "vie.txt"
print(premiere_occurence("lion", fich)) ⟶ 16797
print(premiere_occurence("empereur", fich)) ⟶ 111891
print(premiere_occurence("tigre", fich)) ⟶ -1
```

Contrôler ces réponses en utilisant la fonction find de Python.

On cherche maintenant à savoir combien de fois un mot donné est présent dans un fichier.

Q6) Ecrire une fonction nombre_occurrences(mot, texte) (avec mot et texte de type str) qui renvoie le nombre d'occurrences de mot dans texte. Les occurrences doivent être disjointes, elles ne doivent pas se chevaucher. Vérifier que

```
nombre_occurrences("ata", "taratata") = 1
fich = contenu du fichier "vie.txt"
nombre_occurrences("lion", fich)) = 2
nombre_occurrences("vie", fich)) )= 245
nombre_occurrences("chat", fich)) )= 12
```

Contrôler ces réponses en utilisant la fonction count de Python.

Q7) Calculer la complexité des fonctions premiere_occurrence(mot, texte) et nombre_occurrences(mot, texte) en fonction des paramètres nM = len(mot) et nT = len(texte).

Q8) Ecrire une fonction toutes_phrase(mot, texte) qui retrouve toutes les phrases du texte contenant mot. Retrouver les phrases de "Une vie" contenant les mots "lion", "chat". Les phrases sont séparées par un point.
Comment peut-on faire pour avoir uniquement les phrases contenant le mot lion ou chat (et pas allions ou chatte, par exemple) ?