FICHE DE L'ATELIER 17 : GESTION DES FICHIERS (4 pages)

ENREGISTRER : méthode open avec un argument « w » comme write

Les deux premières lignes de vos programmes	`#!/usr/bin/env python` *Précise que votre fichier.py est un code Python* `# -*- coding: utf-8 -*-` Précise que votre fichier .py est encodé en UTF-8
Création ou ouverture d'un fichier	`obj_fichier = open("mon_fichier.txt","w")` L'objet-fichier se nomme ici : **obj_fichier** Le nom du fichier-texte enregistré sera : « **mon_fichier.txt** » Si le fichier-texte n'existe pas, il sera créé automatiquement. ATTENTION : **Votre nouveau contenu va écraser le précédent**.

Ecrire dans le fichier	L'écriture est séquentielle : on commence au début et on rajoute les éléments à la suite les uns des autres.
	`obj_fichier.write("Première entrée.")`
	`obj_fichier.write("Seconde entrée.")`
	Ces lignes vont enregistrer :
	`Première entrée.Seconde entrée.`
	Pour enregistrer les passages à la ligne, il faut utiliser **\n** dans la chaîne.
	`obj_fichier.write("Première entrée.\n")`
	`obj_fichier.write("Seconde entrée.\n")`
	Ces lignes vont enregistrer :
	`Première entrée.`
	`Seconde entrée.`
	`...`

Fermeture du fichier	`obj_fichier.close()`

RAJOUTER DES CHOSES : méthode open avec un argument « a » comme append

Ouverture d'un fichier	`obj_fichier = open("mon_fichier.txt","a")` L'objet-fichier se nomme ici : **obj_fichier** Le nom du fichier-texte enregistré sera : « **mon_fichier.txt** » Si le fichier-texte n'existe pas, cela déclenche une erreur. Pensez à utiliser un TRY EXCEPT pour contrer ce problème.
Écrire dans le fichier	Aucune différence avec la méthode « w » write.

LIRE UN FICHIER : méthode open avec un argument « r » comme read

Ouverture d'un fichier **open()**	`obj_fichier = open("mon_fichier.txt","r")` L'objet-fichier se nomme ici : **obj_fichier** Le nom du fichier-texte enregistré sera : « **mon_fichier.txt** » Si le fichier-texte n'existe pas, cela déclenche une erreur. Pensez à utiliser un TRY EXCEPT pour contrer ce problème.
Lire un fichier avec la méthode **read()**	Cette méthode renvoie une chaîne de caractère contenant TOUT le fichier. le_contenu = `obj_fichier.read()` Place tout le contenu du fichier dans le string le_contenu. print(`obj_fichier.read()`) Affiche tout le contenu du fichier sur la console. le_contenu = `obj_fichier.read(10)` Place les 10 prochains caractères du fichier dans le string le_contenu. Pour repartir au début (ou à une autre position en mettant autre chose que 0) ; `obj_fichier.seek(0)`

Lire un fichier avec la méthode
 	
readlines()
	Cette méthode renvoie une liste dont chaque élément est l'une des lignes de votre fichier..
	
	le_contenu = `obj_fichier.readlines()`
	Place tout le contenu du fichier dans la liste le_contenu.
	
	print(`obj_fichier.readlines()`)
	Affiche la liste entre [ ] (qui contient les lignes du fichier séparées par des virgules).
	Par exemple :
	`['-- Ouverture --\n' ,'Première entrée.\n' ,'Deuxième entrée.\n' ,'Troisième entrée.\n' ,'-- Fermeture --\n' ,'-- Ouverture en rajout --\n' ,'4e entrée.\n' ,'5e entrée.\n' ,'6e entrée.\n' ,'-- Fermeture --\n']`
	
	Pour connaître la longueur de la liste (et donc ici le nombre de lignes :
	`nbr_lignes = len (le_contenu)`
	
	Pour afficher la première ligne, QUI PORTE LE NUMERO 0 :
	`print(le_contenu[0])`
	
	Pour afficher les lignes une à une :
	`obj_fichier = open("mon_fichier.txt","r")`
	`lecture = obj_fichier.readlines()`
	`for ligne in lecture:`
	`  print(ligne, end='')`
	`obj_fichier.close()`

Lire un fichier avec la méthode
 	
readline()
	Cette méthode renvoie la ligne correspond à l'endroit où pointe le lecteur de fichier.
	
	`obj_fichier = open("mon_fichier.txt","r")`
	`print(obj_fichier.readline())`
	`print(obj_fichier.readline())`
	`obj_fichier.close()`
	
	Ce code va vous afficher les deux premières lignes si elles existent car le fichier pointe naturellement sur la position 0 lors de l'ouverture du fichier.
	
	Pour lire l'intégralité d'un fichier avec un readline :
	
	`#!/usr/bin/env python`
	`# -*- coding: utf-8 -*-`
	`obj_fichier = open("mon_fichier.txt","r")`
	`while 1: # La condition est donc toujours True`
	`  ligne = obj_fichier.readline() # On tente de lire la ligne`
	`  if not(ligne): # Si ligne n'existe pas, elle contient False`
	`  break # On sort de la boucle`
	`  else: # Sinon, c'est que ligne contient quelque chose`
	`  print(ligne)`
	`obj_fichier.close()`

CHEMINS ET REPERTOIRE

Pour descendre dans un sous-répertoire	L'adresse relative par rapport au répertoire courant est par exemple : **`"dossier_2/"`** Le nom du fichier est précédé du nom du sous-répertoire : **`"dossier_2/fichier_1.txt"`**
Pour remonter dans le répertoire maître	L'adresse relative par rapport au répertoire courant est alors : **`"../"`** Le nom du fichier est précédé de deux points : **`"../fichier_1.txt"`**

Pour connaître le répertoire courant avec la méthode
 	
getcwd()
	Cette méthode se trouve dans le module os. Elle signifie get current working directory.
	On obtient en retour une chaîne de caractères contenant **l'adresse absolue** du répertoire.
	Par exemple : **`"C:/dossier_travail/dossier_1/dossier_2/fichier_2.txt"`**
	Le premier slash indique que c'est une adresse absolue depuis la racine du disque C.
	
	`import os`
	`rep_courant = os.getcwd()`

Pour lister le contenu d'un répertoire avec la méthode
 	
listdir()
	Si on veut connaître le contenu en fichier et dossier d'un répertoire, il faut utiliser la méthode listdir(rep_courant) en fournissant donc en argument l'adresse du répertoire voulu.
	
	La méthode renvoie alors une liste [ ] des différents objets trouvés.
	
	`import os`
	`rep_courant = os.getcwd()`
	`liste = os.`**`listdir`**`(rep_courant)`
	
	Puisqu'il s'agit d'une liste, on peut en lire le contenu avec une boucle **for x in liste:**
	
	`for x in liste:`
	`  print(x)`
	
	Les éléments x sont juste des strings contenant le nom de l'élément.

TRIER LES ELEMENTS D'UN REPERTOIRE

Pour détecter si un élément est un dossier :

isdir()

Pour détecter si un élément est un fichier :

isfile()

Une fois les éléments x d'un dossier-répertoire connus à l'aide de listdir(), on peut utiliser deux fonctions contenues dans la bibliothèque os. Elles renvoient respectivement True si x est dossier ou un fichier.

Exemple :

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os

rep_courant = os.getcwd()
liste = os.listdir(rep_courant)

for x in liste:
    if os.path.isfile(x): # True si x est un fichier
        print("FICHIER : "+x)
    elif os.path.isdir(x): # True si x est un dossier
        print("DOSSIER : "+x)
```

Pour trier les fichiers en fonction de leurs extensions :

endswith()

Cette méthode des strings renvoie True si le string x finit par la chaîne fournie en argument. Très pratique pour différencier les extensions.
Ainsi x.endswith('.txt') renvoie True avec « mon_fichier.txt » et False avec « fichier.doc ».

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
rep_courant = os.getcwd()
liste = os.listdir(rep_courant)
for x in liste:
    if os.path.isfile(x): # True si x est un fichier
        if x.endswith('.txt'): # True si x est un fichier txt
            print("FICHIER TXT : "+x)
        elif x.endswith('.jpg'): # True si x est un fichier jpg
            print("FICHIER IMAGE JPEG : "+x)
        elif x.endswith('.png'):
            print("FICHIER IMAGE PNG : "+x)
        else:
            print("FICHIER D'UN AUTRE TYPE : "+x)
```