Fichierstextes

8 septembre 2019

1 Utilisation de fichiers textes avec Python

Pour pourvoir accéder un fichier texte en python, il suffit d'indiquer son chemin d'accès dans un script python avec la fonction **open**. On refermera ce fichier avec la fonction **close**.

[ ]:
```
fichier = open('exemple.txt','r') # ouverture du fichier qui est dans le même␣
 ↪répertoire que mon script.
contenu = fichier.read()# la fonction open attend deux arguments : le nomn du␣
 ↪cfichier à ouvrir (ici exemple.txt),
print(contenu) # et le mode d'ouverture 'r' signifie lecture ici.
fichier.close()
```

** Les 3 modes principaux ** :

-**mode 'r'** : on est en lecture. Sur notre exemple, on affecte à la variable (contenu) l'intégralité du fichier, à l'aide de la méthode **read**. L'impression de cette variable donne l'intégralité du fichier.

-**mode 'w'** : le mode écriture, avec **écrasement** des données préalables.

_**mode 'a'** : le mode ajout, le texte entré est alors ajouté.

[ ]:
```
# Exemple pour le mode 'w'

fichier =  open('exemple.txt','w') #exemple pour le mode 'w'
contenu = fichier.write("C'était vraiment un texte très basique !") # Change le␣
 ↪contenu du fichier
fichier = open('exemple.txt','r') #On recharge en lecture le ficher
contenu= fichier.read() #Lecture
print(contenu) #visualisation
```

[ ]:
```
#exemple pour le mode 'a'
fichier =  open('exemple.txt','a') #exemple pour le mode 'a'
contenu = fichier.write(" Donc là, on ne va pas effacer la phrase précédente ?")␣
 ↪# Change le contenu du fichier
fichier.close()
fichier = open('exemple.txt','r') #On recharge en lecture le ficher
contenu= fichier.read() #Lecture
print(contenu)
```

Au-delà de la mise en forme du texte, on voit bien le résultat. Pour aller à la ligne on utilise le marqueur ** n**

[ ]:
```
fichier =  open('exemple.txt','a')
contenu = fichier.write("\n Donc là, on ne va pas effacer la phrase précédente ?
 ↪") # Change le contenu du fichier
fichier.close()
fichier = open('exemple.txt','r') #On recharge en lecture le ficher
contenu= fichier.read() #Lecture
print(contenu)
```

On peut aussi utiliser la méthode **readlines** pour lire les lignes indépendamment les unes des autres, la méthode **readlines** transférant toutes les autres lignes dans une liste des chaînes.

[ ]:
```
fichier =  open('exemple.txt','r')
contenu = fichier.readlines()
print(contenu)
```

Comment entrer des valeurs numériques en écriture ?

Tout ce que vous rentrez en écriture est du type chaîne de caractères, soit <> en python.

Si l'on veut rentrer des valeurs numériques, il faut d'abord les convertir en chaîne de caractères.

[ ]:
```
fichier =  open('exemple.txt','a')
contenu = fichier.write(str(8))
fichier.close()
fichier = open('exemple.txt','r')
contenu=fichier.read()
print(contenu)
```

Il existe aussi un module du nom de **pickle** qui permet de gérer différents types de données dans un fichier.

[ ]:
```
import pickle
a,b,c,d=1.2,5,'complexe',[1,2,'réel']
contenu = open('données_test','wb')
pickle.dump(a,contenu)
pickle.dump(b,contenu)
pickle.dump(c,contenu)
pickle.dump(d,contenu)
contenu.close()
contenu = open('données_test','rb')
e=pickle.load(contenu)
f=pickle.load(contenu)
g=pickle.load(contenu)
h=pickle.load(contenu)
print(e,type(e))
print(f,type(f))
print(g,type(g))
print(h,type(h))
```

Ici, on utilise **'wb'** et **'rb'** comme méthode, signifiant que l'on écrit en binaire et non en format texte (d'ailleurs notre fichier n'a pas d'extension).

La fonction **dump** du module **pickle** attend deux arguments : la variable et l'objet fichier où l'on travaille.

La fonction **load** du module **pickle** restitue chaque variable avec son type.
Pas belle la vie???