

PROGRAMMATION EN PYTHON

LES FICHIERS SOUS PYTHON

Fatiha BENDAIDA

DWFS 1

2024/2025

FICHIER SOUS PYTHON

○ Introduction :

Dans tous les langages de programmation, on utilise la notion de fichier. Qui désigne un ensemble d'informations enregistrées sur un support (clé usb, disque dur, etc.).

Dans la majorité des langages de programmation, on distingue deux types de fichiers :

- Les fichiers **textes** : les informations sont sous un format texte (.txt, .docs, ...) qui est **lisible** par **n'importe quel éditeur de texte**.
- Les fichiers **binaires**: les informations **ne sont lisibles** que par le **programme qui les a conçus** (.pdf, .jpg, ...).



MODULE Os

- Le module **OS** en python fournit des **fonctions d'interaction** avec le système d'exploitation. Il permet d'effectuer des opérations courantes liées au système d'exploitation. Il est **indépendant** par rapport au **système d'exploitation** de la machine. Ce qui signifie que ce module peut fonctionner sur n'importe quel système d'exploitation.
- **Opérations sur les fichiers et dossiers**
- **Récupérer le répertoire de travail**

```
import os  
print(os.getcwd())
```

```
C:\Users\MSI
```



MODULE Os

○ Changer le répertoire de travail

```
os.chdir('C:/Users/MSI/Desktop')  
# Le dossier de destination doit exister
```

```
print(os.getcwd())
```

C:\Users\MSI\Desktop

- **NB :** le dossier de destination doit exister sinon une exception de type **FileNotFoundError** sera levée??

```
try:  
    os.chdir('C:/Users/MSI/Desktop/abc')  
except IOError:  
    print("le dossier est introuvable!!")
```

le dossier est introuvable!!



GESTION DES EXCEPTIONS

Comme d'autres langages, Python fournit également la gestion d'exceptions à l'aide de **try/except**.

Syntaxe :

```
try:
    code
except type1:
    ce qui doit faire
except type2:
    ce qui doit faire
```

Exemple :

```
def division(a, b):
    try:
        d = a//b
        print(d)
    except ZeroDivisionError:
        print("Division par zero ! ")
```

```
|: def trier(L):
    try:
        L.sort()
        return L
    except:
        print("Veuillez introduire une liste!!!")
```

```
|: trier([3+4,1,5])
```

[1, 5, 7]

```
|: trier(3+4*2)
```

Veuillez introduire une liste!!!

MODULE Os

Opérations sur les fichiers et dossiers

○ Lister le contenu d'un dossier

```
print(os.listdir('C:/Users/MSI/Desktop/cours BTS/1 er année/Python'))
```

```
['3. Cours_Liste_complet.pdf', 'Cours_4_string_dictionnaire (1).pptx', 'fusion.png', 'Introduction En Python.pdf', 'Introduction En Python.pptx', 'Liste-Tuple_par
```

○ Renommer un fichier ou un dossier

os.rename(old,new)

```
os.rename('DWFS', 'DWFS25')
```

○ Créer un dossier

```
os.mkdir('C:/Users/MSI/Desktop/cours BTS/1 er année/Python/Tp Python')
```



MODULE Os

Opérations sur les fichiers et dossiers

○ Création des dossier imbriqués

La méthode **makedirs()** crée plusieurs dossiers intermédiaires dans un chemin. Fondamentalement, cela signifie que vous pouvez créer un chemin contenant des **dossiers imbriqués**.

```
os.makedirs('C:/Users/MSI/Desktop/cours BTS/1 er année/Python/DWFS1/miniprojet')
```



MODULE Os

Opérations sur les fichiers et dossiers

○ Supprimer un fichier ou un dossier

Les fonctions **os.remove()** et **os.rmdir()** sont utilisées pour supprimer des fichiers et des répertoires vides.

Mais sortant d'abord du répertoire à supprimer.

```
os.chdir('C:/Users/MSI/Desktop/cours BTS/1 er année/Python')#je change Le repertoire vers Python
os.rmdir('DWFS1')
```

```
-----
OSError                                Traceback (most recent call last)
Cell In[22], line 2
      1 os.chdir('C:/Users/MSI/Desktop/cours BTS/1 er année/Python')#je change le repertoire vers Python
----> 2 os.rmdir('DWFS1')

OSError: [WinError 145] Le répertoire n'est pas vide: 'DWFS1'
```

```
os.chdir('C:/Users/MSI/Desktop/cours BTS/1 er année/Python/DWFS1')#je change Le repertoire vers DWFS1
os.rmdir('miniprojet')
os.chdir('C:/Users/MSI/Desktop/cours BTS/1 er année/Python')#je change Le repertoire vers Python
os.rmdir('DWFS1')
```


MODULE Os

○ Autres fonctions du module os

Commande	Description
<code>os.path.exists(chemin)</code>	Renvoie True si le chemin existe et False sinon.
<code>os.path.isfile(chemin)</code>	Renvoie True si le chemin est un fichier et False sinon.
<code>os.path.isdir(chemin)</code>	Renvoie True si le chemin est un dossier et False sinon.



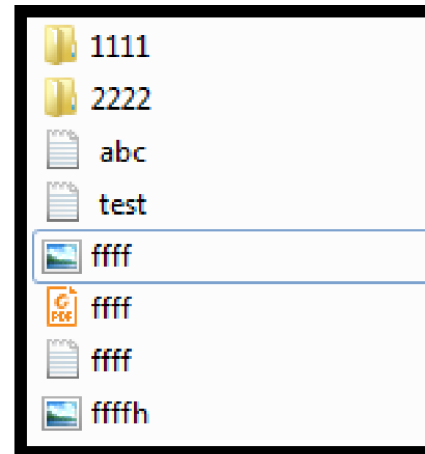
Exercice :

Ecrire une fonction **search_file(chemin,extension)** qui permet de lister tous les fichiers ayant cette extension. La recherche doit aussi portée sur les sous dossiers du chemin d'une manière récursive.

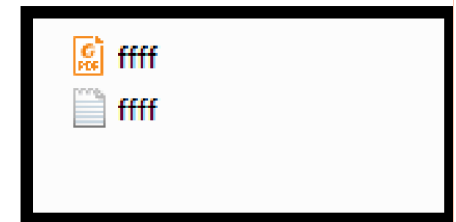
```
chemin='C:/Users/Admin/Desktop/abc'  
extension="txt"  
search_file(chemin,extension)
```

Résultat :

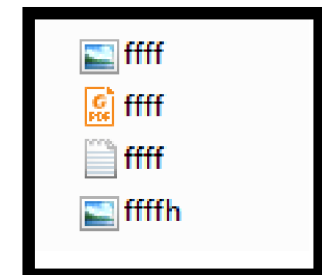
```
C:/Users/Admin/Desktop/abc/ abc.txt  
C:/Users/Admin/Desktop/abc/ test.txt  
C:/Users/Admin/Desktop/abc/1111/ffff.txt  
C:/Users/Admin/Desktop/abc/2222/ffff.txt  
C:/Users/Admin/Desktop/abc/ffff.txt
```



abc

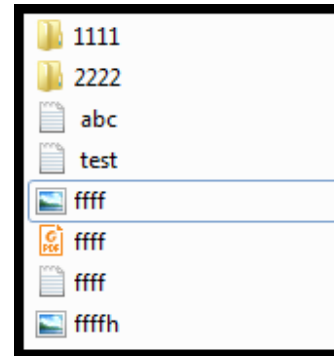


1111

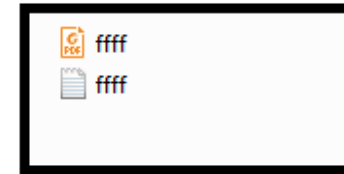


2222

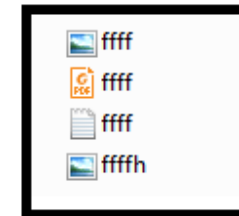
Exercise :



abc



1111



2222

```
import os
def search_file(chemin,extension):
    n=len(extension)
    list_rep=os.listdir(chemin)
    for e in list_rep:
        if e[-n:]==extension:
            print(chemin+"/"+e)
        else:
            search_file(chemin+"/"+e,extension)
```

```
C:/Users/Admin/Desktop/abc/ abc.txt
C:/Users/Admin/Desktop/abc/ test.txt
C:/Users/Admin/Desktop/abc/1111/ffff.txt
C:/Users/Admin/Desktop/abc/2222/ffff.txt
C:/Users/Admin/Desktop/abc/ffff.txt
```



FICHER SOUS PYTHON

○ Introduction :

Les principales manipulations sur un fichier sont :

- **L'ouverture** du fichier
- La **lecture** ou **l'écriture** d'un élément dans un fichier
- La **fermeture** du fichier

○ Ouverture d'un fichier :

Tout fichier doit être ouvert avant de pouvoir accéder à son contenu en **lecture** et **écriture**. L'ouverture d'un fichier est réalisée par la fonction **open** selon la syntaxe suivante :

Variable = **open**(chemin du fichier, **mode**)

mode :

'r' : mode **lecture** seule

'w' : mode **écriture** seule

'a' : mode **ajout**

on ajoute **'b'** si le fichier est **binaire**



FICHIER SOUS PYTHON

Exemples :

```
>>> f1 = open ('test1.txt', "r")  
#ouvrir le fichier test1 qui se trouve en même dossier que le  
fichier du code source.
```

```
>>> f2 = open ("c:/test2.txt" , 'w')  
#ouvrir le fichier test2 qui se trouve dans le répertoire C.
```

```
>>> f3 = open (" c:\\python\\test3.txt" , 'a')  
#ouvrir le fichier test3 qui se trouve dans le dossier python  
du répertoire C.
```

Syntaxe général:

```
f=open(chemin du fichier, mode)  
# blocs d'instructions  
f.close()
```



FICHIER SOUS PYTHON

Remarques :

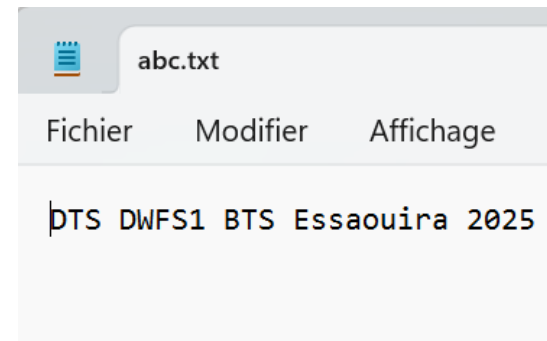
- En mode **lecture**, le fichier doit **exister** sinon une exception de type ***FileNotFoundError*** sera levée.
- En mode **écriture**, Si le fichier n'existe pas, il est **créé** sinon, son contenu est **perdu**.
- En mode **ajout**, si le fichier existe déjà, il sera **étendu**. sinon, il sera **créé**.
- N'oublie pas à la fin de **fermer** le fichier afin que d'autres programmes puissent le lire via la commande :
f1.close() # fermeture du fichier f1

FICHER SOUS PYTHON

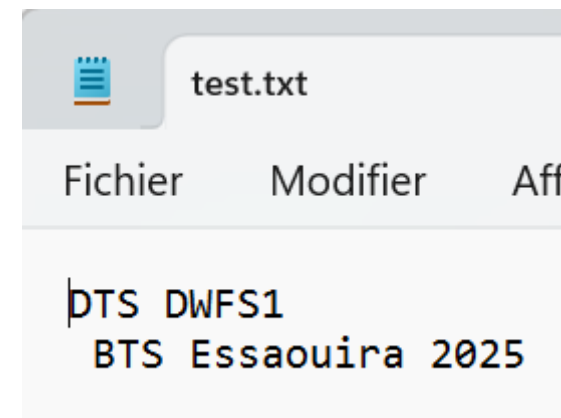
Ecriture dans un fichier

La méthode `write(message)` permet d'écrire la chaîne de caractère *message* dans un fichier. elle retourne le nombre de caractères écrits dans le fichier

```
f=open('abc.txt','w')
N=f.write('DTS DWFS1 BTS Essaouira 2025')
print(N)
f.close()
```



```
f=open('test.txt','w')
N=f.write('DTS DWFS1\n BTS Essaouira 2025\n')
print(N)
f.close()
```

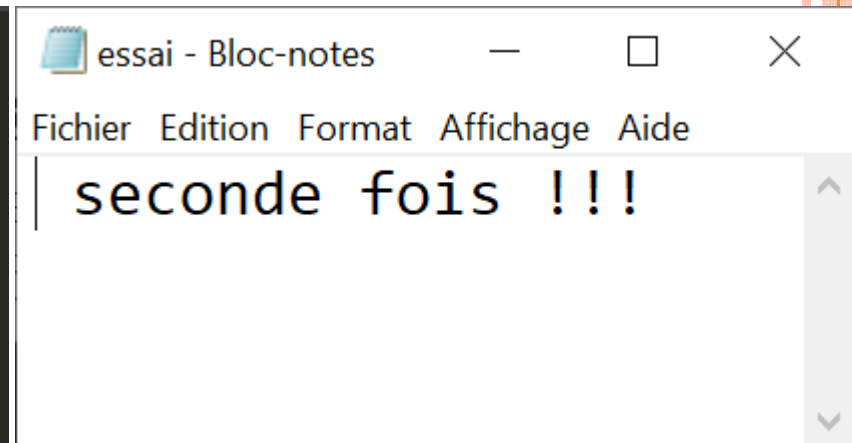


FICHER SOUS PYTHON

Ecriture dans un fichier

- Quelle est le contenu du fichier 'essai.txt' après l'exécution du programme suivant ?

```
f = open ("essai.txt", "w")
f.write (" premiere fois ")
f.close ()
f = open ("essai.txt", "w")
f.write (" seconde fois !!!")
f.close ()
```



Et si on change le deuxième mode en 'a'




FICHIER SOUS PYTHON

Lecture d'un fichier

- **méthode `read(t)`** : Pour lire la quantité **t en octets** à partir de la position déjà atteinte dans le fichier et les retourne en tant que chaîne de caractères. Quand **t** est **omise** ou **négative**, le contenu **tout entier** du fichier est **lu et retourné**.

```
f=open('abc.txt','r')
data1=f.read(2)
f.read(4)
data2=f.read(1)
print(data1+f.read(3)+data2)
f.close()
```



abc.txt		
Fichier	Modifier	Affichage
DTS DWFS1 BTS Essaouira 2025		

DTS1 F

FICHER SOUS PYTHON

Exercice :

- Quel est le résultat de l'exécution du programme ci-dessus ?

```
f = open("essai.txt", "w")
f.write("Bonjour DWFS1, facile n'est-ce pas !!!")
f.close()

f = open("essai.txt", "r")
data1 = f.read(3)
data2 = f.read()
data3 = f.read(4)
print(data1 + "#" + data3)
f.close()
```

Bon#

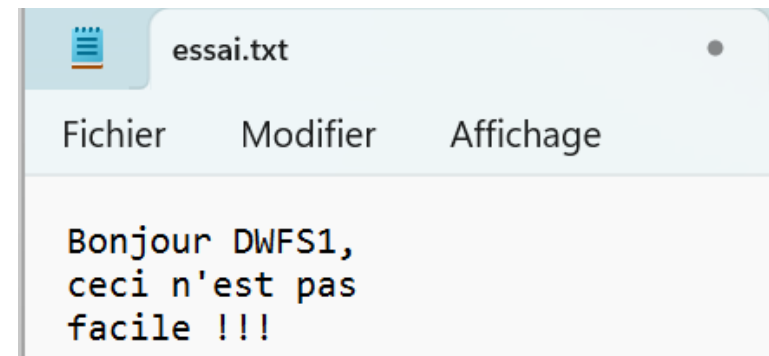


FICHIER SOUS PYTHON

Lecture d'un fichier

- **méthode `readline()`** : lit une seule ligne à partir du fichier.

```
f = open ("essai.txt", "r")  
line = f.readline()  
print(line)  
f.close ()
```



Bonjour DWFS1,

- **méthode `readlines()`** : retourne une liste contenant toutes les lignes du fichier

```
f = open ("essai.txt", "r")  
lines = f.readlines()  
print(lines)  
f.close ()
```

['Bonjour DWFS1,\n', 'ceci n'est pas\n', 'facile !!!']

FICHER SOUS PYTHON

Exercice :

1. Ecrire un programme qui permet de stocker dans un fichier nommé "**diviseurs.txt**" les diviseurs séparés par '#' d'un nombre N lu au clavier.

```
: N=int(input("donner un nombre N : "))
f = open ("diviseurs.txt", "w")
L=[str(i)+'#' for i in range(1,N//2+1) if N%i==0]+[str(N)]
f.write("".join(L))
f.close()
```

donner un nombre N : 24

2. Ecrire un programme qui permet de lire ces nombres à partir du fichier et afficher leurs somme.

```
: f = open ("diviseurs.txt", "r")
ch = f.readline()
L=[int(x) for x in ch.split('#')]
print(sum(L))
f.close()
```

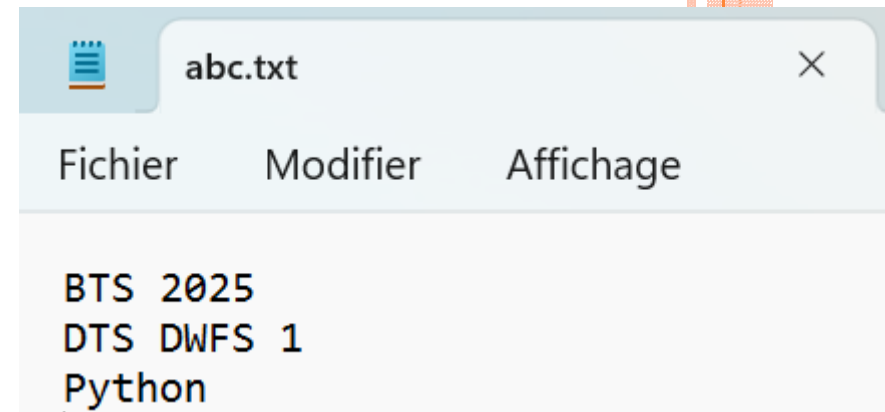
FICHER SOUS PYTHON

Boucle sur un fichier

Remarque : Un fichier texte ouvert en **lecture** possède également les propriétés d'un **itérateur** : on pourra alors **parcourir les lignes** du fichier **par une boucle** :

```
f = open("abc.txt", "r")
for ligne in f:
    print(ligne)
f.close()
```

BTS 2025
DTS DWFS 1
Python



FICHER SOUS PYTHON

Ouverture du fichier avec with

De façon pragmatique, l'instruction **with** permet d'écrire un code sans utiliser l'instruction **close**. Les deux bouts de code suivant sont équivalents :

```
f = open("abc.txt", "r")
for ligne in f:
    print(ligne)
f.close()
```

```
with open("abc.txt", "r") as f:
    for line in f:
        print(line)
```



FICHIER SOUS PYTHON

Exercice : On désire stocker les notes des étudiant d'une classe dans un fichier.

1. Ecrire un programme dans lequel l'utilisateur rentre :

- Le nom du fichier (on exige l'extension '.txt')
- Le nombre des étudiants
- La saisie des noms et notes ($0 \leq \text{note} \leq 20$)
- Stockage sous la forme : **nom \t note \n**

```
file=input("donner le nom du fichier : ")
file=file+".txt"
n=int(input("donner le nombre des étudiants : "))
with open(file,"w") as f :
    for i in range(n):
        name=input("donner le nom de l'étudiant N° %d :"%(i+1))
        note=float(input("donner sa note : "))
        while not(0<=note<=20):
            note=float(input("s.v.p saisir une note entre 0 et 20 : "))
        f.write(name+'\t'+str(note)+'\n')
```

FICHIER SOUS PYTHON

2. On désire maintenant calculer le moyen de la classe des notes contenu dans le fichier précédent.

```
file=input("donner le nom du fichier : ")
file=file+".txt"
with open(file,"r") as f :
    L=f.readlines()
    n=len(L)
    moy=0
    for e in L:
        s=e.split('\t')
        moy+=float(s[1][: -1])
print("la moyenne des notes est :",moy/n)
```

