المدرسة العليا للتكنولوجيا - الصويرة
L'ÉCOLE SUPÉRIEURE DE TECHNOLOGIE – ESSAOUIRA
MARRAKECH
جامعة القاضي عياض
UNIVERSITÉ CADI AYYAD

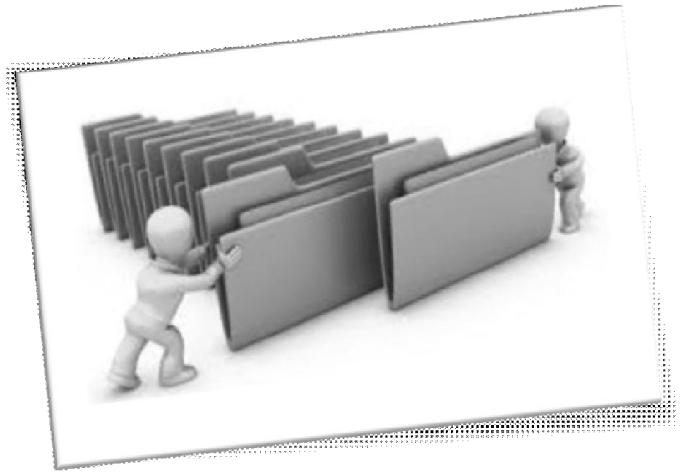
PROGRAMMATION EN PYTHON

DUT IDSD 1

2019/2020

4. LES FICHIERS SOUS PYTHON

PR. MUSTAPHA JOHRI



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, …).

FICHIER 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, mode)

Mode :

'r' : mode lecture seule
'w' : mode écriture seule
'a' : mode ajout
'b' : binaire

FICHIER SOUS PYTHON

- **Introduction :**

Les principales manipulations sur un fichier sont :

Exemples :

```
>>> f1 = open (‘c:\\test1.txt’, "r")
 #ouvrir le fichier test1 en mode lecture seulement
>>> f2 = open ("c:\\test2.txt" , ‘wb’)
 #ouvrir le fichier test1 en mode écriture binaire
>>> f3 = open ("test3.txt" , ‘a’)
#ouvrir le fichier test1 en mode ajout (append)
```

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

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

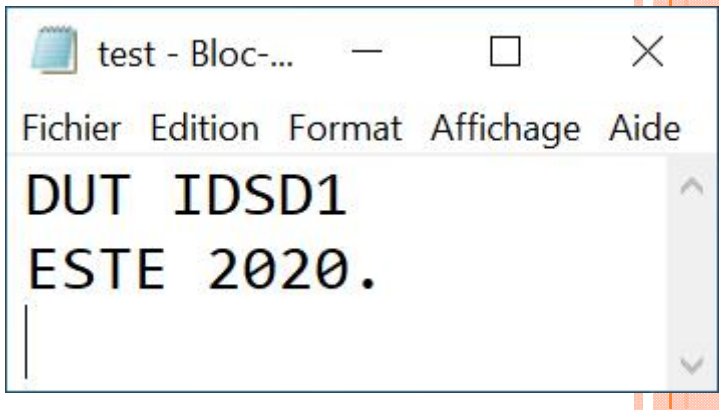
FICHIER SOUS PYTHON

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('DUT IDSD1 ESTE 2021.')
print(N)
f.close()
```


```
f = open (" test.txt ", 'w')
f.write ('DUT IDSD1\nESTE 2020.\n')
f.close ( )
```



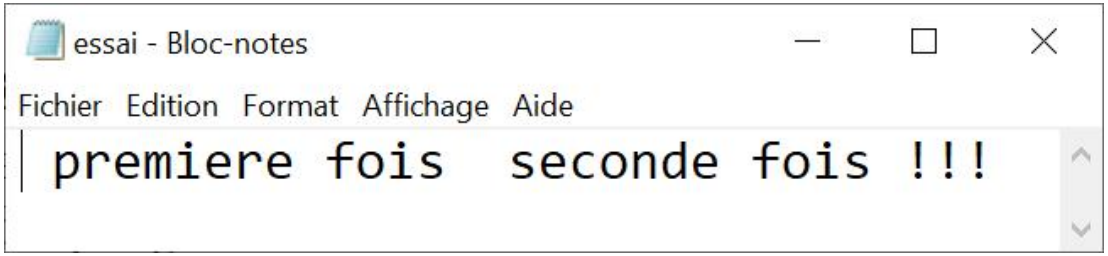
FICHIER SOUS PYTHON

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'

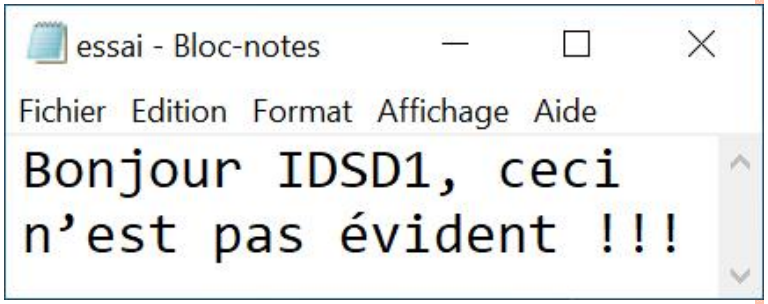


FICHIER SOUS PYTHON

Lecture d'un fichier

- **méthode read(t) :** Pour lire la quantité **t** 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 ("essai.txt", "r")
data1 = f.read(2)
data2 = f.read(4)
print(data1 , " ****" ,data2)
f.close ()
```



Bo **** njou

FICHIER SOUS PYTHON

Exercice 1:

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

```
f = open ("essai.txt", "w")
f.write ("Bonjour IDSD1, ceci n'est pas évident !!!")
f.close ()
f = open ("essai.txt", "r")
data1 = f.read()
data2 = f.read()
print(data1 ,  "****" ,data2)
f.close ()
```

Bonjour IDSD1, ceci n'est pas évident !!! **

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 IDSD1,

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

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

['Bonjour IDSD1,\n', 'ceci n'est pas\n', 'facile !!!']

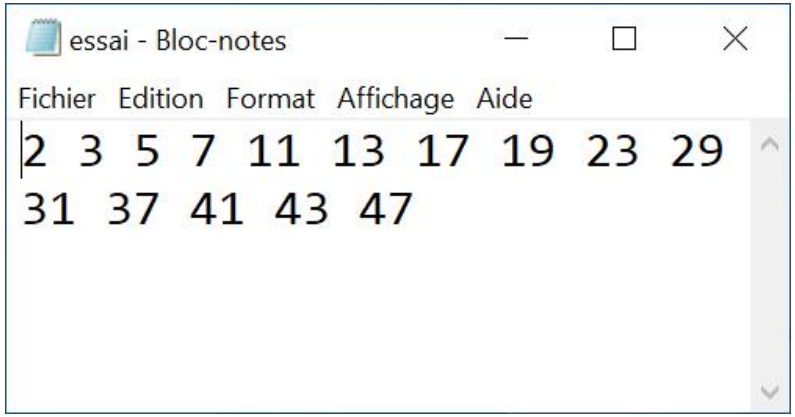
FICHIER SOUS PYTHON

Exercice 2:

1. Ecrire un programme qui permet de stocker dans un fichier les nombres premiers qui sont inférieurs à 50.

```
def premier(n):
    if n<2:
        return False
    for i in range(2,n//2+1):
        if n%i==0:
            return False
    return True

f = open("essai.txt", "w")
for i in range(50):
    if premier(i):
        f.write(str(i)+' ')
f.close()
```



FICHIER SOUS PYTHON

Exercice 2:

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

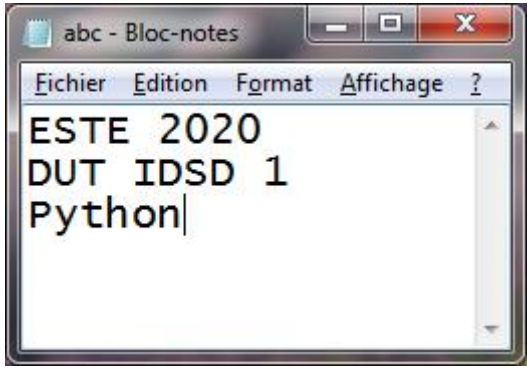
```
f = open("essai.txt", "r")
ch=(f.readline()).strip()
L=ch.split(' ')
print(sum([int(i) for i in L]) )
f.close()
```

FICHIER SOUS PYTHON

Lecture d'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()
```




```
ESTE 2020
DUT IDSD 1
Python
```

FICHIER SOUS PYTHON

Remarque : De façon pragmatique, l'instruction **with** permet d'écrire un code plus court 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 3: On désire stocker les notes des étudiant d'une classe dans un fichier. 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<= note <= 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

Exercice 4: On désire maintenant calculer le moyen de la classe des note 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)
```

FICHIER SOUS PYTHON

Enregistrer des objets dans des fichiers

Grâce au module **pickle**, on peut enregistrer n'importe quel objet et le récupérer par la suite, au prochain lancement du programme.

Tout d'abord, il faut importer le module pickle.

```
import pickle
```

FICHIER SOUS PYTHON

Enregistrer un objet dans un fichier

On utilise la méthode **dump** du p**ickle** pour enregistrer l'objet. Son emploi est :

Syntaxe :

pickle.dump(objet , fichier)

```
from pickle import*
texte = "Python pour IDSD"
quantiteFournitures = {"cahiers":134,
                       "stylos":{"rouge":41,"bleu":74},
                       "gommes": 85}
fournitures = ["cahier", "crayon", "stylo", "trousse", "gomme"]
with open("forniture.dat","wb") as f:
    dump(texte, f)
    dump(quantiteFournitures, f)
    dump(fournitures, f)
```

FICHIER SOUS PYTHON

Classe Unpickler : Récupérer nos objets enregistrés

- Pour lire les objets contenues dans notre fichier, nous utilisons la méthode **load** de notre **module pickle**. Elle renvoie le premier objet qui a été lu (s'il y en a plusieurs, il faut l'appeler plusieurs fois).

```
with open('donnees.dat', “rb“) as f:
        objet = pickle.load(f)
        # Lecture du premier objet contenu dans le fichier...
```

FICHIER SOUS PYTHON

Récupérer nos objets enregistrés

```
from pickle import*
with open("forniture.dat","rb") as f:
    texte=load(f)
    quantiteFournitures=load(f)
    fournitures=load(f)
    print(texte,quantiteFournitures,fournitures,sep='\n')
```

Resultat

Python pour IDSD
{'cahiers': 134, 'stylos': {'rouge': 41, 'bleu': 74}, 'gommes': 85}
['cahier', 'crayon', 'stylo', 'trousse', 'gomme']

FICHIER SOUS PYTHON

Exercice 5:

On cherche a stocker dans un fichier une liste des valeurs L et une formule sous forme d'une fonction.

Exemple : L=[0,1,2,3] et formule $f(x)=x^5-3x^3+1$

1. Ecrire un programme qui permet de stocker les données dans un fichiers nommé 'save.p'

```
from pickle import*
L=[i for i in range(10)]
def f(x):
    return x**5-3*x**3+1
with open('save.p','wb') as fich:
    dump(f,fich)
    dump(L,fich)
```

FICHIER SOUS PYTHON

Exercice 5:

2. Ecrire un programme qui permet de lire les données du fichiers ‘save.p’ et appliquer la formule aux éléments de la liste.

```
from pickle import*
with open('save.p','rb') as fich:
    f=load(fich)
    L=load(fich)
    print(list(map(f,L)))
```

Résultat :

[1, -1, 9, 163, 833, 2751, 7129, 15779, 31233, 56863]

GESTION DES EXCEPTION SOUS PYTHO

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 ! ")
```

```
from pickle import *
try:
    f = open("sam.dat", "rb")
    f.close()
except FileNotFoundError:
    print("Attention!! le fichier n'existe pas")
```

Module Os

Opérations sur les fichiers et dossiers

- En Python, on peut changer le répertoire de travail courant. Pour cela, vous devez utiliser une fonction du module **os**
- **Se placer dans un dossier**

```
import os
os.chdir('C:/Users/Admin/Desktop/abc')
# le dossier de destination abc doit exister
```

- **Récupérer le répertoire de travail**

```
chemin = os.getcwd()
```

```
C:\Users\Admin\Desktop\abc
```

MODULE OS

Opérations sur les fichiers et dossiers

- **Lister le contenu d'un dossier**

```
liste = os.listdir('C:/Users/Admin/Desktop/abc')
```

```
['fich1.txt','fich2.txt','donnees.dat']
```

- **Renommer un fichier ou un dossier**

```
os.rename('ancien_nom.txt', 'nouveau_nom.txt')
```

```
os.rename('fich1.txt', 'texte1.txt')
liste = os.listdir('C:/Users/Admin/Desktop/abc')
print(liste)
```

```
['texte1.txt','fich2.txt','donnees.dat']
```

Module Os

Opérations sur les fichiers et dossiers

- **Supprimer un fichier**

```
os.remove('fichier.txt')
```

- **Créer un dossier**

```
os.mkdir('/chemin/dossier_vide/')
```

- **Vérifier si le fichier ou dossier existe**

```
if os.path.exists('/chemin/fichier.txt'):
        print('Le fichier existe.')
```

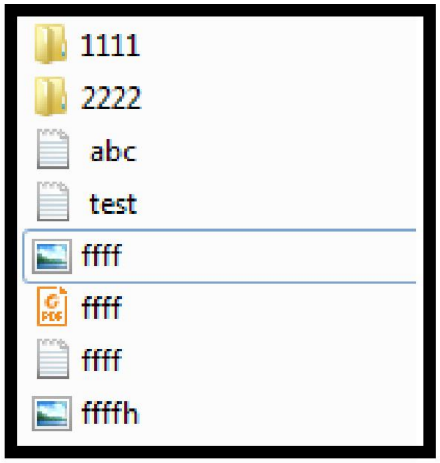
- **Vérifier s'il s'agit d'un fichier**

```
if os.path.isfile('/chemin/fichier.txt'):
        print("L'élément est un fichier.")
```

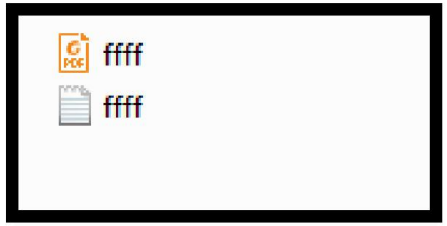
Exercice 6:

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

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



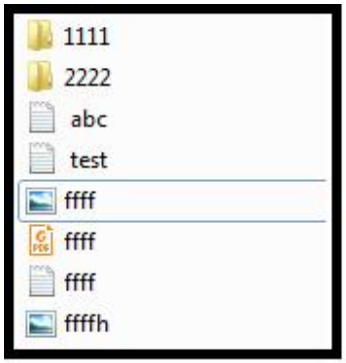
abc



1111

2222

Exercice 6:



abc

1111

2222

```
import os
def search_file(rep,ext):
    liste=os.listdir(rep)
    for e in liste:
        if os.path.isfile(e):
            if e[-3:]==ext:
                print(rep+"/"+e)
        else:
            search_file(rep+"/"+e,ext)
```

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