# Fitxers

Un arxiu o fitxer informàtic és un conjunt de bytes que són emmagatzemats en un dispositiu(HD,FD,SD,Pendrive). Un arxiu és identificat per un nom i la ruta(path) de la carpeta que el conté. Als arxius informàtics se'ls anomena així perquè són els equivalents digitals dels arxius escrits en expedients, targetes, llibretes, paper o microfitxes de l'entorn d'oficina tradicional.

A Python hi ha 3 tipus de fitxers: fitxers de text, fitxers binaris amb buffer i fitxers binaris raws

Buffering(https://medium.com/@bramblexu/understand-the-buffer-policy-in-python-78e91e7759ca)

https://docs.python.org/3/library/io.html

Mireu-vos els mètodes open, read, readline, readlines i write

## Fitxers de text

Els fitxers de text treballen a nivell de caràcter, si els obrim amb el notepad per exemple, són llegibles. La seva estructura és pràcticament com si fos una string, fins al punt que podem llegir tot el seu contingut, ficar-lo dintre d'una String i tractar-lo com una string.

Aquest tipus de fitxers són els que es que farem servir habitualment com Administradors de Sistemes, sobretot en Linux. Fitxers de configuració de serveis, fitxers de logs, ... els hi podrem aplicar expressions regulars, importar-los com a XML, CSV, JSON, YAML, ... i tractar-los com a llistes, diccionaris, ...

Amb aquest tipus de fitxers és molt important saber com estat codificats (utf8, ASCII, ...)

http://python-notes.curiousefficiency.org/en/latest/python3/text_file_processing.html

https://python.readthedocs.io/en/stable/library/codecs.html

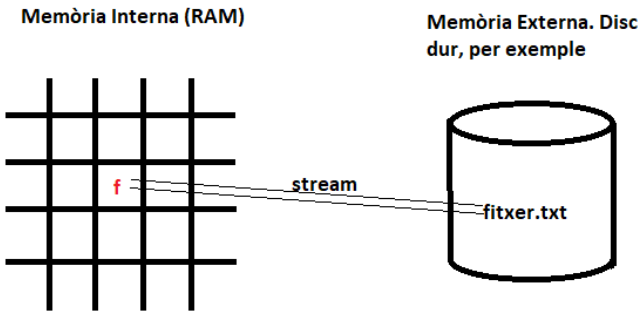
Quan treballem amb fitxers tindrem una sèrie de passos que s'hauran de fer.

### Primer pas. Obrir el fitxer. Open()

Aquest pas el que fa és connectar una variable de python amb un fitxer extern. Aquesta és coneix tradicionalment com "stream". A la instrucció següent

```
f =  open("elmeufitxer.txt")
```

estem obrint el fitxer extern "elmeufitxer.txt" amb la variable f(stream). Just en aquest pas es crea com una espècie de cananonada(tubería, pipeline) entre f(és una variable: memòria interna) i "elemeufitxer.txt"(memòria externa: HD, SSD, PenDrive, ...)




Els fitxers de text es poden obrir de les següents formes(mode)

| Mode | Descripció |
|---|---|
| x | Només serveix per crear el fitxer, si ja existeix dóna error. El fitxer s'obre per escriptura. |
| r | **Read**. Obre el fitxer per a lectura. El punter és posiciona al Principi del fitxer. Si no indiquem el mode al fer open, és la manera predeterminada. |
| r+ | Obre el fitxer per a lectura i escriptura. El punter de lectura és posiciona al Principi del fitxer. I el punter d'escriptura al final |
| w | **Write**. Si el fitxer ja existeix, trunca a zero la longitud(es carrega el contingut anterior). Si no existeix crea un fitxer de text. En tots dos casos s'obre per a escriptura. El punter és posiciona al Principi del fitxer. |
| w+ | Ídem w però també poden llegir. El punter és posiciona al Principi de l'fitxer |
| a | **Append**. Obre fitxer per a escriptura. És crearà el fitxer si no existeix. El punter és posiciona a la fi del fitxer. |
| a+ | Obre fitxer per a lectura i escriptura. És crearà el fitxer si no existeix. El punter és posiciona a la fi del fitxer. Si volem posicionar-nos per llegir dades hem de fer servir seek() |

Esquema amb el que es pot fer realment en cada mode

https://mkyong.com/python/python-difference-between-r-w-and-a-in-open/

**Que és això del punter?**

Quan s'obre un fitxer i per exemple l'obrim per Read i el punter és a l'inici, vol dir que quan fem la primera lectura la farà des del principi del fitxer.

Quan obrim un fitxer per Append, el punter és al final, això significa, que si faig la primera escriptura, es farà al final del fitxer.

## Segon pas. Llegim i/o modifiquem el fitxer

Farem el que calgui amb el fitxer ja sigui llegir( mètodes read(),readline(),readlines(),...), escriure(mètode write(),writelines(),...), posicionar-nos(seek()), o altres com tell(),readable(),writeable(), ...

Per exemple si vull escriure “Hola” seguit d’un salt de línia al stream f que abans hem obert per escriptura, farem:

```
f=open(“fitxer.txt”,mode=”w”)
f.write(“Hola\n”)
....
```

### Tercer pas. Tanquem el fitxer. Close()

Fins que no tanquem el fitxer no es desa el contingut del stream a “disc”. Per tant si no hem tancat el fitxer i es va el llum, tots els writes que hem fet es perden.

També passa, que al tancar el fitxer s’allibera la memòria que ocupa el stream. En l’exemple anterior

```
f=open(“fitxer.txt”,mode=”w”)
f.write(“Hola\n”)
f.close()# Es passa el contingut de f a “fitxer.txt” i s’allibera la memòria que ocupa f
```

Quan s’acaba un programa de forma correcta, si no s’ha fet el close, python el fa automàticament, però no us confieu, no tots els llenguatges fan el mateix.

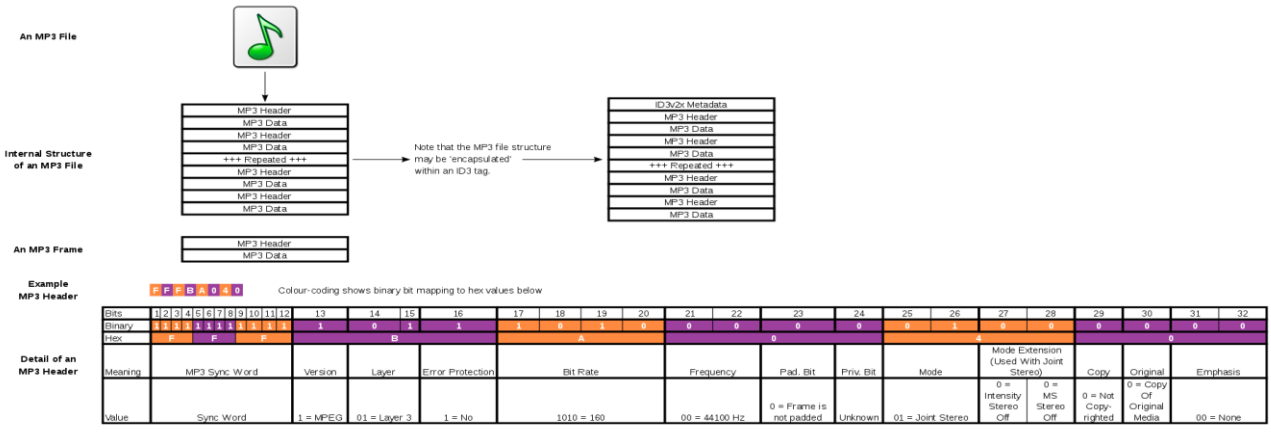
Simplificació del procés. with ... as ...

Per poder simplificar totes els passos python té una estructura que va molt bé, el with, que **tanca el fitxer automàticament.**

```
with open(“elmeufitxer.txt”,mode=”w”) as f:
  f.write(“Hola\n”)
```

## Fitxers Binaris

Els fitxers binaris treballen a nivell de bit(realment byte), per poder obrir-los necessitem saber quina és la seva estructura interna. Per exemple, per obrir un mp3 necessitem saber com es guarda un mp3, sinó amb el nostre programa serem incapaços de entendre el seu contingut.




També és normal en aquests tipus de fitxers, que la seva estructura no estigui publicada(siguin propietat d’una empresa), i necessitis un producte comercial per obrir-lo.  Per exemple una Base de Dades de SQL Server(.mdf, .ldf, .ndf), per obrir-la necessites el producte MS SQL Server.

Altra exemple és un fitxer de .docx(Word), encara que el seu contingut sembla Text, no és text, és binari, no s'enten si l'obres amb el bloc de notes. En concret és un fitxer ZIP, el podeu descomprimir, i veure que internament són molts fitxers.

Com a programadors de Python, el més habitual per accedir a fitxers binaris és importar alguna llibreria ja feta, ja que com veieu amb l'exemple de MP3, senzill no és. Aquí teniu un enllaç de com tractar els fitxers binaris, byte a byte.

https://www.devdungeon.com/content/working-binary-data-python

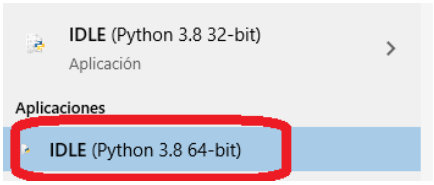
A Python hi ha moltes llibreries per accedir i/o modificar les dades d'un mp3, una d'elles és eyed3. Per poder instal·lar-la segueix els següents passos. Compte, aquesta llibreria es fa servir per llegir i/o modificar les metadates(ID3 Tag) del MP3(autor, títol, album, ..., ) no per reproduir mp3.

- Obre una sessió **com a administrador** al CMD
- Actualitza el programa pip amb la següent comanda

  python -m pip install --upgrade pip

- Afegeix la lliberia eyed3. Aquesta llibreria s'afegeix al Python de 64 bits, per tant, si fas servir el Idle, utilitza el de 64 bits, sinó no la trobarà. Si no et funcionés el pip, seguarement et faltin les rutes de Python al Path del SO(https://datatofish.com/add-python-to-windows-path/ )

  pip install eyed3




- Dintre del teu codi de Python importa la llibreria amb la comanda de sempre

  import eyed3

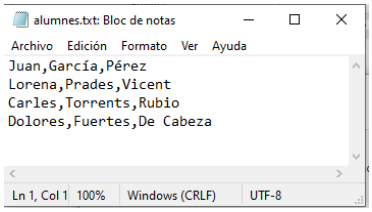
- Ja pots fer-la servir, en el següent enllaç tens com fer-la servir

  https://eyed3.readthedocs.io/en/latest/

# Exemples

## Exemple1. Lectura d'un fitxer de text línia a línia

Volem fer un programa que vagi llegint un fitxer de text línia a línia i les vagi mostrant per pantalla conforme les va llegint. He creat una carpeta dades i dintre hi ha els fitxers de dades, en aquest cas faré servir un fitxer que es diu "alumnes.txt"( l'he creat amb el notepad de Windows, això generarà problemes)




El meu programa és. A programació sempre s'han de fer servir rutes relatives. (../, ./)

```
ruta = "./dades/alumnes.txt"
with open(ruta,"r") as f:
    linia = f.readline()
    while(linia):
        print(linia)
        linia = f.readline()
```

I el resultat és

```
========== RESTART: C:/Users/
Juan,GarcÃ-a,PÃ©rez

Lorena,Prades,Vicent

Carles,Torrents,Rubio

Dolores,Fuertes,De Cabeza
>>>
```

Mostra les dades separades per un INTRO addicional i els accents no li agraden.

Si us fixeu en la imatge del bloc de notes  a la part de sota posa Windows(CRLF) i utf-8. Això significa que el fitxer està codificat en utf-8, però si mirem quina taula de codis està fent servir python per defecte( sobretot si teniu Windows) veiem el següent:

```
>>> import locale
>>> print(locale.getpreferredencoding(False))
cp1252
```

No està fent servir la codificació unicode utf-8 sinó una ANSI cp1252, per tant es fa un embolics amb els accents i els posa malament, per tant quan obrim un arxiu, per no tenir aquests problemes, haurem de dir com està codificat, com volem que es codifiqui si és per escriptura.

Per altra banda, fa un salt de línia addicional. Això és perquè Python ve del mon Linux i el Salt de línia és LF, i en Windows és CRLF, així al fer servir readline a la variable línia, per exemple la de "Juan,García,Pérez", realment té "Juan,García,Pérez\n" i per això al fer el print mostra una línia de més. Això es soluciona eliminant aquest \n de la línia llegida amb un replace.

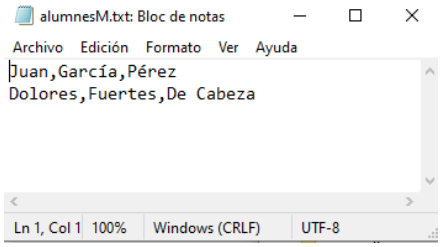
```
ruta = "./dades/alumnes.txt"
with open(ruta,"r",encoding="utf-8") as f:
    linia = f.readline().replace("\n","")
    while(linia):
        print(linia)
        linia = f.readline().replace("\n","")
```

## Exemple 2. Llegir d'un fitxer i escriure en un altre

Volem generar un nou fitxer anomenat "Alumnes fins a la M.txt" que contindrà tots els alumnes que tenen un primer cognom que comenci per una lletra que estigui abans de la M o per la M.

```
rutaAlumnes = "./dades/alumnes.txt"
rutaAlumnesM= "./dades/alumnesM.txt"
#Podem fer un with amb més d'un fitxer
#Obrim el fitxer alumnes per lectura
#Obrim el fitxer alumnesM per exriptura
with open(rutaAlumnes,"r",encoding="utf-8") as alumnes, open(rutaAlumnesM,"w",encoding="utf-8") as alumnesM:
    linia = alumnes.readline().replace("\n","")
    while(linia):
        cognom1 = linia.split(",")[1] #Trocejem la línia per les comes, i agafen la segona columna  [1]
        if cognom1[0]<="M": #Suposem que el cognom começa per majúscula, com ha de ser.
          alumnesM.write(linia+"\n") # Li afegeixo a la nova línia un salt de línia, el write no el fa per defecte
        linia = alumnes.readline().replace("\n","")
```

Amb les dades del fitxer alumnes.txt, us haurà creat el nou fitxer alumnesM.txt




## Exemple 3. Fitxer de log

Farem servir un fitxer de log, on cada 20 segons guardarem l'ús de la CPU

```
import psutil #s'ha d'instal·lar amb el pip
import time
from datetime import datetime

ruta = "./dades/log.txt"

while(True):
    cpu = psutil.cpu_percent()
    ara = datetime.now()
    with open(ruta,mode="a",encoding="utf-8") as log:
        log.write("{0:%Y/%m/%d} CPU:{1:.2f}%\n".format(ara,cpu))
    time.sleep(20)
#Obrim i tanquem el fitxer cada volta del while, perquè sinó no podríem
#veure els canvis, ja que no es faria el close
#Com que fem servir un fitxer que ja existeix, i volem afegir línies
#fem servir mode append
```

# Mètodes de la Classe File

https://www.w3schools.com/python/python_ref_file.asp

# Pràctica(2.5 punts). Tots els exercicis valen el mateix

Cotxes.txt:

Seat 600
Lanborghini Diablo
Ferrari Testarosa
Renault 14
Renault Fuego
Seat 1430
Renault 8
Renault Alpine
Renault 5
Seat 127
Renault 4
Renault 12

Seat 1500
Seat 131 Supermirafiori

1. Ens mostri per pantalla un únic missatge amb el contingut del fitxer cotxes.txt

2. Ens mostri per pantalla tots els Renaults.

3. Generi un fitxer amb tots els Seat, anomenat «Seat.txt»

4. Generi un nou fitxer per a cada marca amb els cotxes corresponenst a la marca: «Seat.txt», «Renault.txt», «Lanborgini.txt», «Ferrari.txt», … Suposarem que només tindrem cotxes d'aquestes 4 marques.

5. Ídem anterior. Suposarem que tindrem cotxes de diferents marques, però no sabem d'entrada quines marques són. La primera paraula serà sempre la marca. En aquest cas. Hauràs de fer un with dintre d'un altra with, el de dintre serà el d'escriptura amb ruta variable. Hauràs de fer servir append.