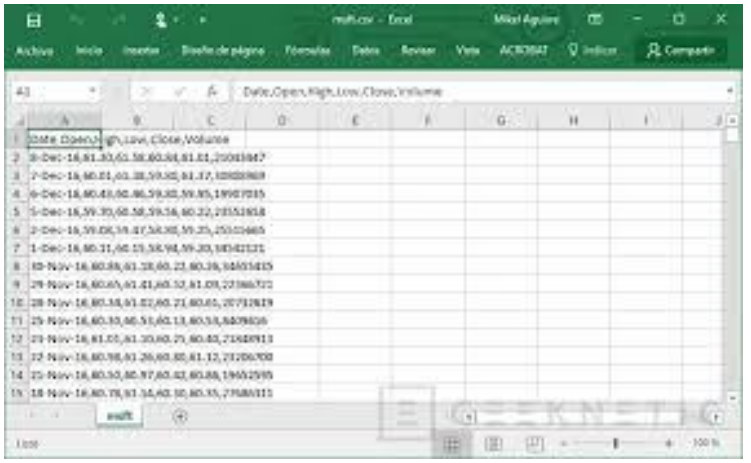
Programació amb Python

Unió Europea
Fons social europeu

Manejo Ficheros

- Archivos de texto
- Archivos binarios



Que es un fichero ?

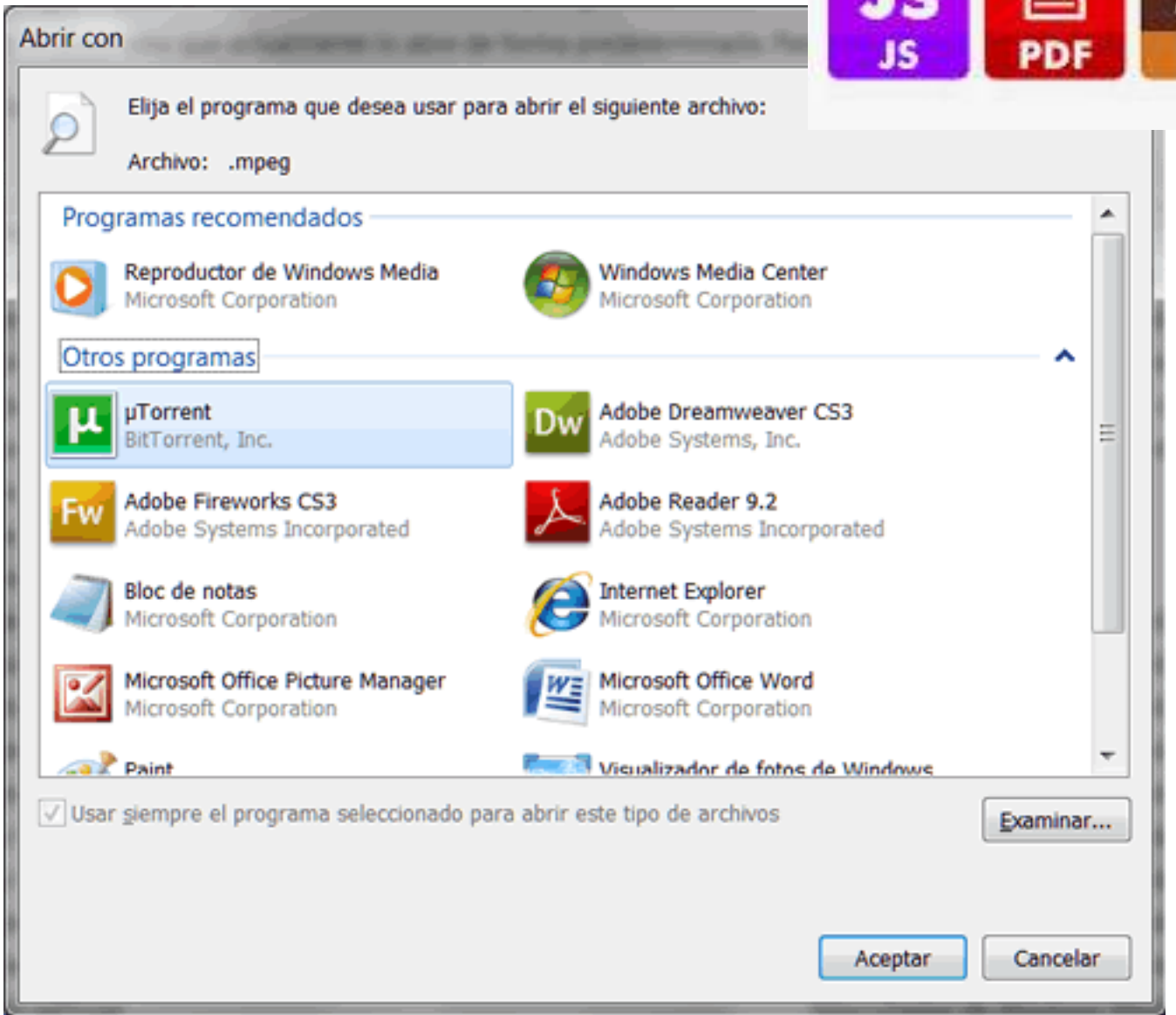
Primeramente, ¿qué es un fichero? Un fichero es un conjunto de bits almacenados en un dispositivo de memoria persistente, normalmente un disco duro.

Por cierto, los ficheros se conocen también como archivos informáticos porque son equivalentes digitales de los archivos escritos por ejemplo en expedientes, tarjetas o libretas que encontraríamos en una oficina tradicional.

- nombre
- extensión o tipo
- ubicación o ruta

Tipos
- de texto codificación (ASCII, UTF-8, UNICODE)
- binarios


JPG
Ps
PSD
MP4
PNG
</>
HTML
{;}
CSS
Js
JS
PDF
Ai
AI
Id
ID
<?>
PHP
TIFF


Abrir con
Elija el programa que desea usar para abrir el siguiente archivo:
Archivo: .mpeg
Programas recomendados
Reproductor de Windows Media
Microsoft Corporation
Windows Media Center
Microsoft Corporation
Otros programas
µTorrent
BitTorrent, Inc.
Adobe Dreamweaver CS3
Adobe Systems, Inc.
Adobe Fireworks CS3
Adobe Systems Incorporated
Adobe Reader 9.2
Adobe Systems Incorporated
Bloc de notas
Microsoft Corporation
Internet Explorer
Microsoft Corporation
Microsoft Office Picture Manager
Microsoft Corporation
Microsoft Office Word
Microsoft Corporation
Paint
Visualizador de fotos de Windows
Usar siempre el programa seleccionado para abrir este tipo de archivos
Examinar...
Aceptar
Cancelar

Identificar ficheros de tipo texto

Al abrirlos con un editor de texto (*) …

1. …puedo ver que el contenido es legible:
   - números y letras de los alfabetos europeos
   - caracteres matemáticos básicos : <>/=+-.
   - signos del lenguaje habitual ¿?!;.()[]{}

2. … suele dividirse en líneas (no és obligatorio) CRLF = \n o siguen un patrón visible de organización de datos.

(*) por ejemplo Notepad o Notepad++…

Texto I / O

Text I / O espera y produce objetos. `str`

La forma más fácil de crear un flujo de texto es `open()` especificando opcionalmente una codificación:

```
f = open("myfile.txt", "r", encoding="utf-8")
```

Los flujos de texto en memoria también están disponibles como `StringIO` objetos:

```
f = io.StringIO("some initial text data")
```

La API de flujo de texto se describe en detalle en la documentación de `TextIOBase`.

https://docs.python.org/3/library/io.html?highlight=open#io.open

Ficheros de texto

Hay varias formas de abrir un fichero, la más común es utilizando la función *open* del módulo *io*.

Creación y escritura

```
from io import open

texto = "Una línea con texto\nOtra línea con texto"

# Ruta donde crearemos el fichero, w indica escritura
# (puntero al principio)
fichero = open('fichero.txt','w')

# Escribimos el texto
fichero.write(texto)

# Cerramos el fichero
fichero.close()
```

Práctica P01. Ficheros

Ejercicio 1 :

```
from io import open

nombre_fichero = '.\\dat\\blacklist.dat'

fichero = open (nombre_fichero, 'w')
fichero.write("malparit")
fichero.close()
```

- Funciona este fragmento de código si la carpeta dat no existe ?
- Y si el fichero blacklist.dat no existe ?
- Si puede crear el fichero, ábrelo con Notepad++, és UTF-8 ?
- Si grabas dos palabras en lugar de una, ¿ Como queda el fichero ?
- Lo puedes rectificar ?

Práctica P01. Ficheros

Ejercicio 2 :

Crea con un editor, un fichero que se llame indice.txt y contenga 10 líneas cada una de ellas que empiece con un número de tema y siga con una descripción.

Créalo dentro de la carpeta cursopy\dat para que esté accesible a tu programa.

1. Traspasando el Top-Secret de la Ufología
2. Contacto Alienígena
3. Identidad Extraterrestre
4. La Física y Ufo.
5. Lugares Magnetizados
6. Colisión Ufo
7. Sobre nuestro Planeta
8. Abducciones
9. La Invasión silenciosa
10. Curaciones Ufo

Ejemplo para CopyPaste

Práctica P00. Tipos de Ficheros

Comentad en clase cuales de estos tipos de ficheros son de texto y cuales son binarios.

Lectura

```
from io import open

# Ruta donde leeremos el fichero, r
# indica lectura(por defecto ya es r)

fichero = open('fichero.txt','r')

# Lectura completa
texto = fichero.read()

# Cerramos el fichero
fichero.close()

print(texto)
```

Ejercicio 3 :

Lee el fichero que has creado en el ejercicio anterior y muestra su contenido por pantalla.

Podemos usar el método readlines() del fichero para generar una lista con las líneas:

Código Resultado

```
from io import open
fichero = open('fichero.txt','r')

# Leempos creando una lista de líneas
texto = fichero.readlines()

fichero.close()
print(texto)
```

Ejercicio 4: Aplica estas instrucciones a tu programa y estudia como funcionan.

También se puede leer un fichero utilizando la instrucción estándar *with* de la siguiente forma:

Código Resultado

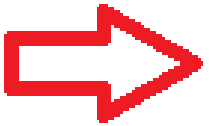
```
with open("fichero.txt", "r") as fichero:
    for linea in fichero:
        print(linea)
```

```
fiche = open("per.txt", "r")

texto =fiche.read()
print (type(texto))

texto =fiche.readlines()
print (type(texto))

texto =fiche.readline()
print (type(texto))
```



```
<class 'str'>
<class 'list'>
<class 'str'>
```

Según la forma en que leemos el fichero, obtenemos un tipo de datos u otro.

Ejercicio 5: Con un editor de texto, crea un fichero con el texto aquí incluido.

Léelo como te parezca mejor, pero calcula y responde si cada una de estas palabras aparece o no en el texto [llibertat, ajudar, ostatges, lliure]

L'advocació de Maria de la Mercè està lligada a la tasca carismàtica de l'ordre religiosa mercedària, nascuda a Barcelona amb sant Pere Nolasc, el seu fundador, mercader original, naturalitzat ciutadà barceloní. Va saber descobrir els dolors dels cristians captius en poder de musulmans, i es va obstinar, amb un grup de companys, primer amb el patrimoni personal, i després recollint almoines, en exercir el ministeri de la caritat eximia: ajudar, visitar i redimir captius. Quan faltaven diners per comprar la seva llibertat, rescatant de les masmorres africanes, s'obligaven tots a quedar en ostatges, esperant que arribessin els seus companys amb els diners previst. Si això no succeïa en el moment oportú, estaven disposats a lliurar la seva pròpia vida.

Extender o Añadir a la cola

Este modo nos permite añadir datos al final de un fichero:

```
from io import open

# Ruta donde leeremos el fichero, a indica extensión (puntero al final)
fichero = open('fichero.txt','a')

fichero.write('\nOtra línea más abajo del todo')

fichero.close()
```

La variante 'a+' permite crear el fichero si no existe:

```
fichero = open('fichero_inventado.txt','a+')
```

Mode	Descripción
'r'	Este es el modo por defecto. Abre el archivo para su lectura.
'w'	Este modo abre el archivo para escribir. Si el archivo no existe, crea un nuevo archivo. Si el archivo existe, trunca el archivo.
'x'	Crea un nuevo archivo. Si el archivo ya existe, la operación falla.
'a'	Abrir archivo en modo de añadir. Si el archivo no existe, crea un nuevo archivo.
't'	Este es el modo por defecto. Se abre en modo texto.
'b'	Esto se abre en modo binario.
'+'	Esto abrirá un archivo para leer y escribir (actualizar)

Sr.No.	Attribute & Description
1	**file.closed** Returns true if file is closed, false otherwise.
2	**file.mode** Returns access mode with which file was opened.
3	**file.name** Returns name of the file.
4	**file.softspace** Returns false if space explicitly required with print, true otherwise.

Métodos con descripción
archivo.close () Cierra el archivo. Un archivo cerrado no se puede leer ni escribir más.
archivo.flush () vacía el búfer interno, como fflush de stdio.
file.fileno () Devuelve el descriptor de archivo entero que utiliza la implementación subyacente para solicitar operaciones de E / S al sistema operativo.
file.next () Devuelve la siguiente línea del archivo cada vez que se llama.
Devuelve True si el archivo está conectado a un dispositivo tty (-like), sino False.

file.read ([tamaño])
Lee como máximo bytes de tamaño del archivo (menos si la lectura alcanza el valor EOF antes de obtener bytes de tamaño).

file.readline ([tamaño])
Lee una línea completa del archivo. Un carácter de nueva línea final se mantiene en la cadena.

file.readlines ([sizehint])
Lee hasta EOF usando readline () y devuelve una lista que contiene las líneas. Si el argumento opcional sizehint está presente, en lugar de leer hasta EOF, se leen líneas completas que suman aproximadamente bytes de sizehint (posiblemente después del redondeo a un tamaño de búfer interno).

file.seek (offset [, whence])
Establece la posición actual del archivo.

archivo.tell ()
Devuelve la posición actual del archivo.

Librería os (operating system)

```
import os
```

Elimina un fichero

```
os.remove("demofile.txt")
```

Elimina una carpeta

```
os.rmdir("myfolder")
```

```
# Create directory
dirName = 'tempDir'

try:
    # Create target Directory
    os.mkdir(dirName)
    print("Directory " , dirName ,  " Created ")
except FileExistsError:
    print("Directory " , dirName ,  " already exists")
```

```
# Create target Directory if don't exist
if not os.path.exists(dirName):
    os.mkdir(dirName)
    print("Directory " , dirName ,  " Created ")
else:
    print("Directory " , dirName ,  " already exists")
```

Práctica P02

Con lo que has estudiado hasta ahora, realiza las siguientes acciones :

- Copia y pega este índice en un fichero de texto con la extension .txt

- Lee e imprime el fichero que has creado.

- Crea una subcarpeta con nombre old y crea allí una copia del archivo. Trabajaremos sobre el original.

- Renumera el capítulo 11 como capítulo 12 y los puntos 1 a 11 como a-k.

Capítulo 11. Manejo de archivos
11.1. Cerrar un archivo
11.2. Ejemplo de procesamiento de archivos
11.3. Modo de apertura de los archivos
11.4. Escribir en un archivo
11.5. Agregar información a un archivo
11.6. Manipular un archivo en forma binaria
11.7. Persistencia de datos
11.8. Directorios
11.9. Resumen
11.10. Ejercicios
11.11. Apéndice

Práctica P03

Con lo que has estudiado hasta ahora, realiza las siguientes acciones :

- Copia y pega estos datos en un fichero con la extensión.csv

- Lee el fichero.

- Imprime todas las líneas en columnas separadas por un tabulador: “Any Codi Municipi Edats Homes Dones Total”

-Crea una copia del archivo y marca con un asterisco adicional al final *, todos los registros que tienen el valor de hombres mayor que el de mujeres. Si el valor de hombres es menor, escribe solo un separador adicional.

Any;Codi;Municipi;Edats;Homes;Dones;Total
2017;250019; Abella de la Conca; De 0 anys;0;0;0
2017;250019; Abella de la Conca; D'1 any;0;1;1
2017;250019; Abella de la Conca; De 2 anys;1;1;2
2017;250019; Abella de la Conca; De 3 anys;0;0;0
2017;250019; Abella de la Conca; De 4 anys;3;0;3
2017;250019; Abella de la Conca; De 5 anys;2;1;3
2017;250019; Abella de la Conca; De 6 anys;0;0;0
2017;250019; Abella de la Conca; De 7 anys;1;0;1

I / O binario

Las E / S binarias (también llamadas *E / S en búfer*) esperan objetos similares a bytes y producen `bytes` objetos. No se realiza ninguna codificación, decodificación o traducción de nueva línea. Esta categoría de flujos puede utilizarse para todo tipo de datos que no sean de texto, y también cuando se desea un control manual sobre el manejo de los datos de texto.

La forma más fácil de crear un flujo binario es con `open()` la `'b'` de la cadena modo:

```
f = open("myfile.jpg", "rb")
```

Los flujos binarios en memoria también están disponibles como `BytesIO` objetos:

```
f = io.BytesIO(b"some initial binary data: \x00\x01")
```

La API de flujo binario se describe en detalle en los documentos de `BufferedIOBase`.

Otros módulos de la biblioteca pueden proporcionar formas adicionales de crear secuencias de texto o binarias. Ver `socket.socket.makefile()` por ejemplo.

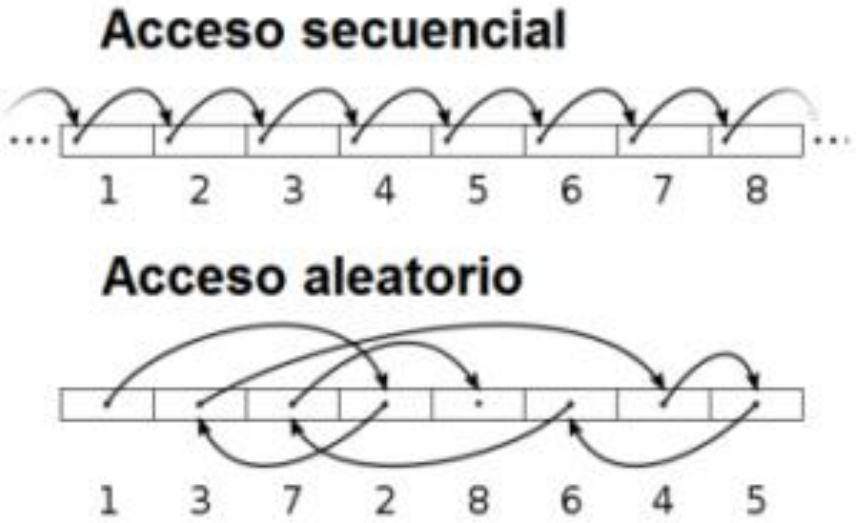
Raw I / O

La E / S *sin* procesar (también llamada *E / S sin búfer*) se usa generalmente como un bloque de construcción de bajo nivel para flujos binarios y de texto; rara vez es útil manipular directamente un flujo en bruto desde el código de usuario. Sin embargo, puede crear una secuencia sin formato abriendo un archivo en modo binario con el búfer deshabilitado:

```
f = open("myfile.jpg", "rb", buffering=0)
```

La API de flujo sin formato se describe en detalle en los documentos de `RawIOBase`.

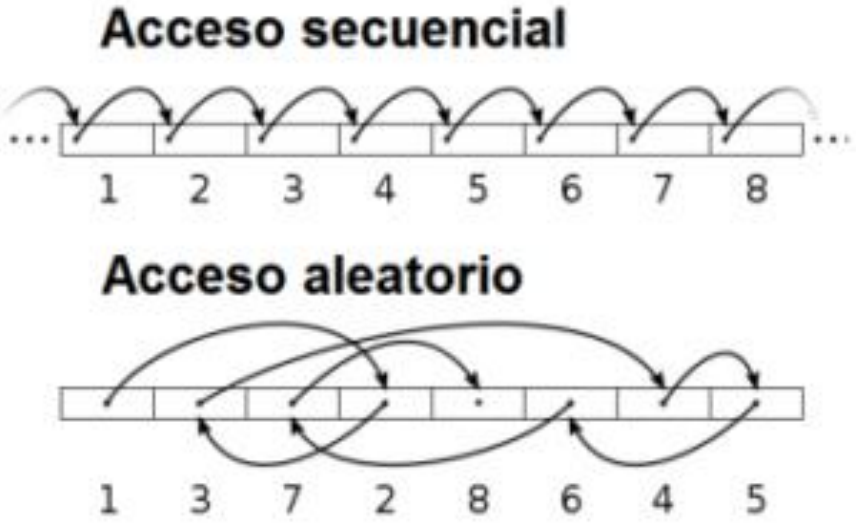
Acceso directo a los datos



Puntero del fichero

Imaginaros el puntero como si fuera el dedo del ordenador mientras recorre el fichero, igual que nosotros seguimos con el dedo un texto mientras lo leemos y así sabemos por dónde vamos.

El puntero es muy importante, ya que por ejemplo, si se encuentra al principio del fichero y le decimos que guarde datos ahí, si no hay nada perfecto, pero si ya hay datos ¿que ocurrirá? Que los guardaremos encima de otros datos y quizá haremos que el fichero quede inservible. Entonces, ¿si queremos añadir datos al fichero, dónde debería estar el puntero? Pues al final del todo, justo donde no hay nada más.



Lectura con escritura

Se puede abrir un fichero en modo lectura con escritura, pero éste debe existir préviamente. Además por defecto el puntero estará al principio y si escribimos algo sobreescribiremos el contenido actual, así que prestad atención a los saltos de línea y caracteres especiales:

```
# Creamos un fichero de prueba con 4 líneas
fichero = open('fichero2.txt','w')
texto = "Línea 1\nLínea 2\nLínea 3\nLínea 4"
fichero.write(texto)
fichero.close()

# Lo abrimos en lectura con escritura y escribimos algo
fichero = open('fichero2.txt','r+')
fichero.write("0123456")

# Volvemos a ponter el puntero al inicio y leemos hasta el final
fichero.seek(0)
fichero.read()
fichero.close()
```

Manejando el puntero

Es posible posicioar el puntero en el fichero manualmente usando el método *seek* e indicando un número de caracteres para luego leer una cantidad de caracteres con el método *read*:

```
fichero = open('fichero.txt','r')
fichero.seek(0)   # Puntero al principio
fichero.read(10)  # Leemos 10 carácteres
```

Para posicionar el puntero justo al inicio de la segunda línea, podríamos ponerlo justo en la longitud de la primera:

```
fichero = open('fichero.txt','r')
fichero.seek(0)

# Leemos la primera linea y situamos el puntero al principio de la segunda
fichero.seek( len(fichero.readline()) )

# Leemos todo lo que queda del puntero hasta el final
fichero.read()
```

Modificar una línea

Para lograr este fin lo mejor es leer todas las líneas en una lista, modificar la línea en la lista, posicionar el puntero al principio y reescribir de nuevo todas las líneas:

```
fichero = open('fichero2.txt','r+')
texto = fichero.readlines()

# Modificamos la línea que queramos a partir del índice
texto[2] = "Esta es la línea 3 modificada\n"

# Volvemos a ponter el puntero al inicio y reescribimos
fichero.seek(0)
fichero.writelines(texto)
fichero.close()

# Leemos el fichero de nuevo
with open("fichero2.txt", "r") as fichero:
    print(fichero.read())
```