



Clase 8 Archivos



Archivos. Concepto y clasificación. Archivos de texto. Archivos separados por comas (CSV).

Teoria

Introducción...

Hasta ahora se vio manejo de datos con variables y listas. Ambas tienen en común que se guardan en la memoria principal del computador. Esta memoria requiere alimentación eléctrica permanente para conservar su contenido.

Para que estos datos perduren en el tiempo es necesario almacenarlo en dispositivos externos que no requieran energía para su conservación. A través del tiempo se han usado distintas tecnologías con este propósito.

Concepto

Cualquiera sea el dispositivo en el que se almacenen los datos, lo que estará guardando siempre será un **ARCHIVO**

Un archivo es un conjunto de elementos llamados **REGISTRO**, todos del mismo tipo de dato, que se almacena en un dispositivo auxiliar para preservar la información a través del tiempo.

Habitualmente cada registro contiene varios datos referidos a un mismo sujeto. Si el sujeto es una persona, estos datos suelen incluir su nombre, número de documento, fecha de nac, etc.

Si el sujeto es un producto, los datos contendrán el código interno de inventario, su descripción, precio de costo, cantidad en stock, etc.

A cada uno de estos datos se lo denomina **campo**.

fila es registro, campo son las columnas y el archivo es todo, todos los registros tienen los mismos campos



jamás se almacenan en la memoria del computador



Un **ARCHIVO** es un conjunto de registros.



Un **REGISTRO** es un conjunto de campos.



Clasificación de archivos

Segun el **sentido de la transferencia** de datos:

- **Archivos de entrada:** En ellos solo se puede **leer**, pero no es posible grabar datos. datos desde el archivo al computador
- **Archivos de salida:** En ellos solo se puede grabar; no es posible leer. desde del computados al archivo.

se leen // se graban

completada la grabación se puede convertir en un archivo de entrada

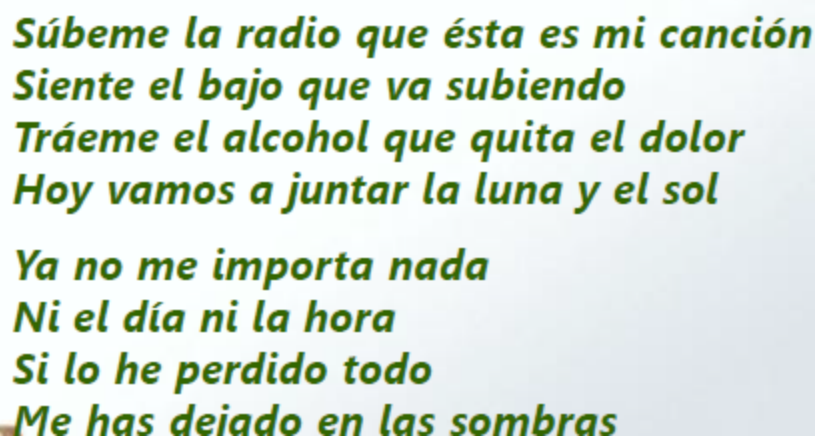
Archivos en Python

En Python se utilizan archivos de textos , es decir archivos formados por **cadena de caracteres**.

Estas cadena contienen solo texto. No hay distintas tipografías , subrayados , negritas , etc. Es lo que se denomina **TEXTO PLANO (solo texto nada raro)**

Los archivos de texto puede ser creados , visualizados o modificados a traves de cualquier editor de texto , como el Block de Notas o IDLE Python. Con extension ".txt"

Ejemplo de un archivo



*Súbeme la radio que ésta es mi canción
Siente el bajo que va subiendo
Tráeme el alcohol que quita el dolor
Hoy vamos a juntar la luna y el sol

Ya no me importa nada
Ni el día ni la hora
Si lo he perdido todo
Me has deiado en las sombras*

registros más largos y registros más cortos. la longitud de registro es variable

Registros

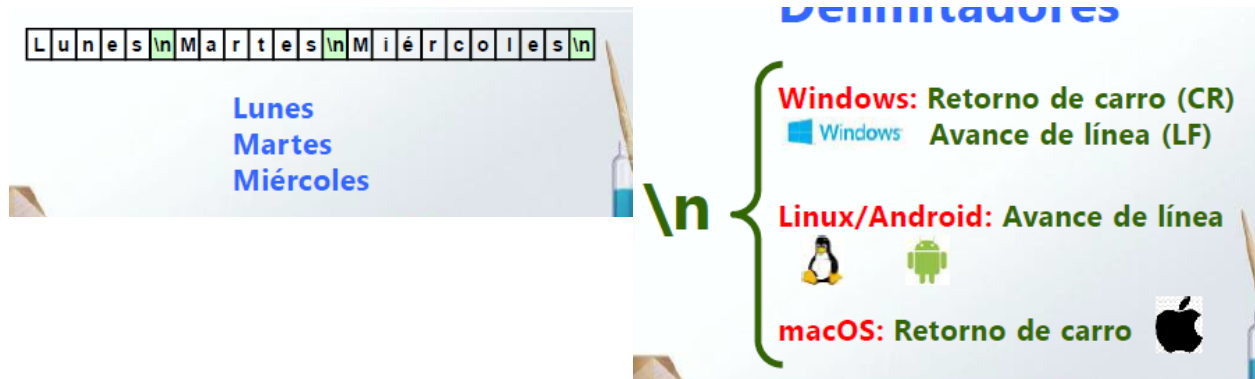
Cada línea de un archivo de texto constituye un **REGISTRO**.

En los archivos de texto la longitud de las líneas es variable , y por lo tanto lo es también la longitud de registro.

Esto obliga a colocar un separador entre cada registro. A este separador se lo conoce como **DELIMITADOR**

Delimitadores

El delimitador que se utiliza es la secuencia de escape `"\n"`, que representa al salto de línea:



Tres etapas en el trabajo con archivos: apertura, procesamiento, cierre

Apertura

- Todo archivo debe ser **abierto** antes de ser utilizado.
- todo archivo reside en un dispositivo auxiliar
- Durante la apertura se establecen canales de comunicación con el dispositivo donde reside el archivo y se reserva memoria para los buffers.
- La apertura se realiza con la función `open()` :

```
<var> = open(<nombre>[, <modo>])
```

`<var>` es la variable que se usará para representar al archivo dentro del programa. Todo el trabajo con el archivo se hará a través de ella.

```
arch = open("datos.txt", "rt")
```

- El nombre puede incluir la ruta deseada

```
arch = open("c:\\nuevo\\datos.txt", "w")
```

```
t")
```



no están los datos del archivo en esta parte

- Si no se incluye ruta al archivo se busca en la misma carpeta donde se encuentra el programa.
- Evitar ruta invalida por el salto de línea ("**\n**")

```
arch = open("c:\nuevo\datos.txt" , "wt")
#Existen tres posibilidades para evitar este error

"1----Usar doble barra invertida---"
arch = open("c:\\nuevo\\datos.txt" , "wt")#La doble contrabarra...
#es tambien una secuencia de escape que representa a una sola...
#barra invertida

"2----Usar una sola barra normal:"
arch = open("c:/nuevo/datos.txt" , "wt")

"3----Declarar la cadena como cruda:"
arch = open(r"c:\nuevo\datos.txt" , "wt")
```

El modo de apertura esta formado por uno o dos caracteres.

El modo básico de apertura(primer) puede ser **r**(read) **w**(write) o **a** (append).

r(read) : Abre el archivo en modo entrada , es decir para lectura solamente

El archivo tiene que existir. En caso contrario se producirá un error.

w(write): Abre el archivo en modo salida , es decir para grabación solamente.

- Si el archivo no existe , sera creado.

- Si el archivo ya existe , sera destruido.

a(append) : Abre el archivo en modo salida , es decir para grabación solamente y agregado de registros.

- Si el archivo no existe , sera creado.
- Si el archivo ya existe , todas las grabaciones se realizaran al final de los datos actuales.

El segundo carácter del modo de apertura es el modificador t (texto).

Si se omite el modo de apertura "rt" (lectura , texto).

Si la apertura fue exitosa , open() devuelve un **objeto archivo** que sera asignado a una variable.

Si ocurre algún problema , se produce una excepción.

Por este motivo todo archivo deberá abrirse siempre dentro de un bloque protegido.

```
try:  
    archivo1 = open("apuntesfinal.txt",[,rr])
```

Los errores que pueden producirse durante la apertura son los sig:

- Nombre invalido
- Archivo de lectura inexistente
- Disco lleno
- Disco protegido contra escritura
- Permisos insuficiente
- Archivo en uso

Cierre

Durante el cierre se revierte todo lo que se hizo en la apertura.

Se clausuran los canales de comunicación con el dispositivo y se liberan los buffers , grabando cualquier registro pendiente que pudiera haber.

Para cerrar un archivo se utiliza el método `close()` de la variable que representa al archivo:

```
arch.close()
```

Debido a la importancia del cierre, se suele realizar en la clausula *finally*.

En muchas implementaciones de Python , intentar cerrar un archivo que no consiguió abrirse provocara un error.

Se soluciona con un bloque protegido dentro del *finally*.

```
finally:
    try:
        arch.close()
    except NameError:
        pass
```

Procesamiento

El procesamiento de un archivo consiste en realizar lecturas y grabaciones sobre el mismo.

Existen dos maneras distintas para grabar y tres para leer.

Todos se realizan con métodos.

Métodos de grabación

- **<arch>.write(<str>)** : Graba <str> en el archivo. El salto de línea debe añadirse manualmente , porque este método no lo agrega.
- **<arch>.writelines(<lista>)** : Graba una lista de cadenas. El salto de línea debe añadirse manualmente a cada elemento de la lista.

EJEMPLO 1:

Leer desde el teclado los datos correspondiente a los alumnos de un curso (legajo y nombre) y grabarlos en un archivo CSV (*comma-separated values*). El fin de datos se indica ingresando un legajo vacío (ENTER).

```
try:
    arch = open("alumnos.txt" , "wt")
    lu = input("LU? (ENTER para terminar):")
    while lu != "":
        nombre = input("Nombre?")
        arch.write(lu+':'+nombre+'\n')
        lu = input("LU? (ENTER para terminar):")
    print("Archivo creado correctamente.")
except OSError as mensaje:
    print("No se puede grabar el archivo:" , mensaje)
finally:
    try:
        arch.close()
    except NameError:
        pass
```

Ejemplo del archivo

```
1042735;Vignale, Juan José
1118693;Garay, Mariela Daiana
1094219;Zanini, Candela Belén
1008752;Blanco, Rodrigo Axel
```

*(Obsérvese que al final del registro no se
agrega punto y coma)*

Excepciones y archivos

- La clase `OSError` usada en el ejemplo anterior es una clase de excepción muy general.
- Agrupa problemas tan diversos como disco lleno , ruta de acceso invalida , permisos insuficiente o archivo inexistente.
- Por esta razón se prefiere usar `OSError` como medida de última recurso , creando antes otros manejadores de excepciones más específicos.
- Se recomienda capturar la excepción **`FileNotFoundError`** antes de `OSError` , tanto en archivos de entrada como de salida.

```
except FileNotFoundError ---> va primero siempre
except OSError as mensaje:---> muy general y es automatizada
    print(mensaje)
```

Metodos de lectura

`<arch>.read([<n>])` : Lee un archivo de texto y devuelve una única cadena de caracteres. Este método lee el archivo entero, lo que puede ser ***muy peligroso***

con archivos grande. Si se incluye el parámetro opcional <n> se lee esa cantidad de caracteres.

- Una posibilidad aceptada es escribir un 1 como parámetro , para leer un carácter por vez.

<arch>.readline(<lista>) : Lee una sola línea del archivo y la devuelve como valor de retorno, o una cadena vacía si no hay mas datos.

- Este sera el método de lectura preferido, ya que permite leer un registro por vez.

Atención

No se permite la carga de archivos completos en **memoria** , ya que constituye una pésima practica de programación.

—seleccionar los primeros 100 registros y trabajar con eso

Se considera **memoria** a un string , lista o otro tipo de estructura de datos.

<arch>.readlines(): Lee archivo entero y lo devuelve como una lista de cadenas.
No se permite.

EJEMPLO 2

Leer el archivo generado en el ejemplo anterior e imprimir por pantalla los datos de aquellos alumnos cuyo numero de legajo sea menor a 1.000.000

```
try:
    arch = open("alumnos.txt" , "rt")
    linea = arch.readline()
    while linea: #MIENTRAS LINEAS SEA DISTINTO DE CADENA VACÍ
A
        lu,nombre = linea.split(';')
        nombre = nombre.rstrip('\n') #ELIMINABA CARAC DE LA D
ERECHA QUE PASABAMOS COMO PARÁMETRO
        if int(lu) < 1000000:
```

```

        print(f"LU:{lu:>7} - Nombre:{nombre}")
    linea = arch.readline()#LEE TODOS LOS REGISTROS
    print("Archivo leído correctamente")
except FileNotFoundError as mensaje:
    print("No se puede abrir el archivo:" , mensaje)
except OSError as mensaje:
    print("No se puede leer el archivo:", mensaje)
finally:
    try:
        arch.close()
    except NameError:
        pass

```

NO PODEMOS ESCRIBIR UN ARCHIVO SI NO LO LEIMOS/CREAMOS

Indicador de posición

¿Qué registro sigue para leer o grabar?

parte de los datos que se guarda en
<archivos>, señalador de un libro



si se llega al final, readline() nos devuelve cadena vacía puntero esta atrás de todo

```
arch.seek(0)
```

leer por segunda vez necesito que vuelva al principio

EJEMPLO 3

Leer el archivo generado en el ejemplo anterior e imprimir por pantalla los datos de aquellos alumnos cuyo numero de legajo sea menor a 1.000.000

Con la instrucción "for" aplicada a un archivo.

```

try:
    arch= open("alumnos.txt","rt")
    for linea in arch:
        lu,nombre = linea.split(";")
        nombre = nombre.rstrip("\n")
        if int(lu) <1000000:
            print(f"LU: {lu:>7} - Nombre: {nombre}")
    print("Archivo leído correctamente")
except FileNotFoundError as mensaje:
    print("No se puede abrir el archivo", mensaje)
except OSError as mensaje:
    print("No se puede abrir el archivo", mensaje)
finally:
    try:
        arch.close()
    except ValueError:
        pass

```

for /-/ no puede ser combinado con lectura read o readline

EJEMPLO 4

Leer un archivo de texto y mostrar la palabra más larga que contenga. Si hay más de una se mostrará cualquiera de ellas

```

try:
    archivo = open("notas.txt", "rt")
    maslarga = " "
    for linea in archivo:
        linea = archivo.strip(";")
        listadepalabras = linea.split( ) #lista de strings
        for palabra in listadepalabras:
            if len(palabra) > len(maslarga):
                maslarga = palabra

```

```

        print(f"La palabra más larga es {maslarga}")
except FileNotFoundError as mensaje:
    print("No se pudo abrir el archivo", mensaje)
except OSError as mensaje:
    print("No se pudo abrir el archivo", mensaje)
finally:
    try:
        archivo.close( )
    except NameError:
        pass
except NameError:
    pass

```

EJEMPLO 5

Convertir a mayúsculas el contenido del archivo "notas.txt"

```

try:
    archivo = open("notas.txt", "rt")
    salida = open("notas2.txt", "wt") #nueva version
    k = 0
    for linea in archivo:
        salida.write(linea.upper())
        k += 1
except FileNotFoundError as mensaje:
    print("No se pudo abrir el archivo", mensaje)
except OSError as mensaje:
    print("No se pudo abrir el archivo", mensaje)
else:
    print("Copia finalizada. Líneas copiadas:", k)
finally:
    try:
        archivo.close( )
    except NameError:
        pass

```

```
except NameError:  
    pass
```

▼ tips

el nombre del archivo es problema nuestro → modulo OS es una herramienta

no debe abrir y cerrarse el archivo por cada registro → arch.seek(0)

no puede ni leerse por campos, l/r siempre en registro

minimizar la cantidad de lecturas que se recorren los archivos

leer un archivo de texto cualquiera e informar cuantas letras de todo el abecedario tiene, 27 contadores en una lista

```
UnicodeDecodeError  
arch = open("datos.txt","rt", encoding="UTF8")  
#solo cuando es necesario
```