

Clase N° 8

Archivos

© Lic. Ricardo Thompson

Introducción

- **Hasta ahora todo el manejo de datos se realizó con variables o con listas.**
- **Ambas tienen en común que se almacenan en la memoria principal del computador.**
- **Esta memoria requiere alimentación eléctrica permanente para conservar su contenido.**

© Lic. Ricardo Thompson

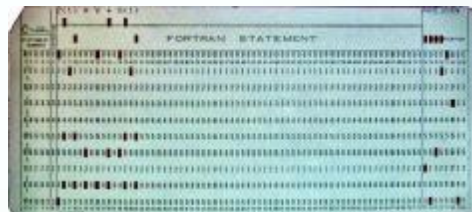
Introducción

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

© Lic. Ricardo Thompson

Introducción

Tarjetas perforadas



Cintas magnéticas

© Lic. Ricardo Thompson

Introducción



Discos



Memorias no volátiles

© Lic. Ricardo Thompson

Concepto

- Cualquiera sea el dispositivo en el que se almacenen los datos, lo que se estará guardando será siempre un **archivo**.
- Un **archivo** es un conjunto de elementos llamados **registros**, todos del mismo tipo de dato, que se almacena en un dispositivo auxiliar para preservar la información a través del tiempo.

© Lic. Ricardo Thompson

Concepto

- 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 nacimiento, domicilio, etc.

© Lic. Ricardo Thompson

Concepto

- 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**.

© Lic. Ricardo Thompson

Concepto

Resumiendo:

- Un **archivo** es un conjunto de registros.
- Un **registro** es un conjunto de campos.

© Lic. Ricardo Thompson

Clasificación de archivos

1. Según el contenido de los registros:

- **Archivos binarios:** En ellos los datos se guardan respetando el mismo formato que tienen en memoria. No se utilizan en Python, aunque se ofrece cierto grado de soporte por compatibilidad.
- **Archivos de texto:** En ellos los datos se almacenan en forma de cadenas de caracteres.

© Lic. Ricardo Thompson

Clasificación de archivos

2. Según el sentido de la transferencia de datos:

- **Archivos de entrada:** En ellos sólo se puede leer; no es posible grabar datos.
- **Archivos de salida:** En ellos sólo se puede grabar, pero no es posible leer.
- **Archivos de entrada/salida:** Se pueden realizar lecturas y grabaciones sin restricciones. Sólo es aplicable a archivos binarios.

© Lic. Ricardo Thompson

Clasificación de archivos

3. Según el método de acceso:

- **Archivos de acceso secuencial:** Para acceder al registro número N es necesario pasar por los $N-1$ registros anteriores.
- **Archivos de acceso directo:** Conociendo el número de registro es posible acceder en forma directa. Son únicamente binarios, por lo que no se tratarán en este curso.

© Lic. Ricardo Thompson

Archivos en Python

- En Python se utilizan fundamentalmente archivos de texto, es decir archivos formados por cadenas de caracteres.
- Estas cadenas contienen sólo texto. No hay distintas tipografías, subrayados, negritas, etc. Es lo que se denomina *texto plano*.

© Lic. Ricardo Thompson

Archivos en Python

- Los archivos de texto pueden ser creados, visualizados o modificados a través de cualquier editor de texto, como el Block de Notas de Windows o el IDLE de Python.
- Generalmente tienen extensión ".txt".

© Lic. Ricardo Thompson

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 dejado en las sombras*

© Lic. Ricardo Thompson

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 también lo es la longitud de registro.
- Ésto obliga a colocar un separador entre cada registro. A este separador se lo conoce como **delimitador**.

© Lic. Ricardo Thompson

Delimitadores

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

```
L u n e s \n M a r t e s \n M i é r c o l e s \n
```

Lunes
Martes
Miércoles

© Lic. Ricardo Thompson

Delimitadores

- El programa grabará "`\n`" como delimitador entre cada registro, pero el carácter efectivamente grabado dependerá del sistema operativo que se esté usando.
- A este proceso de traducción del delimitador se lo conoce como **conversión de datos**, y es exclusivo de los archivos de texto.

© Lic. Ricardo Thompson

Delimitadores

`\n`

Windows: Retorno de carro (CR)
Avance de línea (LF)

Linux/Android: Avance de línea

macOS: Retorno de carro

© Lic. Ricardo Thompson

Las tres etapas en el trabajo con archivos

- **Apertura**
- **Procesamiento**
- **Cierre**

© Lic. Ricardo Thompson

Apertura

- Todo archivo debe ser *abierto* antes de ser utilizado.
- Durante la apertura se establecen canales de comunicación con el dispositivo donde reside el archivo y se reserva memoria para los *buffers*.

© Lic. Ricardo Thompson

Apertura

- 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")
```

© Lic. Ricardo Thompson

Apertura

- El **nombre** puede incluir la ruta deseada.

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

- Si no se incluye ruta al archivo se busca en la carpeta donde se encuentra el programa.

© Lic. Ricardo Thompson

Apertura

- **ATENCIÓN:** Esta ruta es inválida, por el salto de línea ("**\n**") incluido en ella.

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

- Para evitar este error existen tres posibilidades.

© Lic. Ricardo Thompson

Apertura

1. Usar *doble barra invertida*:

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

- La doble contrabarra es también una secuencia de escape que representa a una sola barra invertida.

© Lic. Ricardo Thompson

Apertura

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")
```

© Lic. Ricardo Thompson

Apertura

- El **modo de apertura** está formado por uno o dos caracteres.
- El primero es el modo básico de apertura y puede ser **r** (*read*), **w** (*write*) o **a** (*append*).

© Lic. Ricardo Thompson

Apertura

- **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.

© Lic. Ricardo Thompson

Apertura

- **w (*write*)**: Abre el archivo en modo salida, es decir para *grabación solamente*.
- Si el archivo no existe, será creado.
- Si el archivo ya existe, será destruido.

© Lic. Ricardo Thompson

Apertura

- **a (*append*)**: Abre el archivo en modo salida, es decir para grabación solamente y *agregado de registros*.
- Si el archivo no existe, será creado.
- Si el archivo ya existe, todas las grabaciones se realizarán al final de los datos actuales.

© Lic. Ricardo Thompson

Apertura

- El segundo carácter del modo de apertura es un modificador, y puede ser **b** (*binario*) o **t** (*texto*).
- Si se omite este modificador se asume "t" (*texto*).
- Si se omite todo el modo de apertura se asume "rt" (*lectura, texto*).

© Lic. Ricardo Thompson

Apertura

- Si la apertura fue exitosa, `open()` devuelve un **objeto archivo** que será 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*.

© Lic. Ricardo Thompson

Apertura

- **Los errores que pueden producirse durante la apertura son diversos:**
 - **Nombre inválido**
 - **Archivo de lectura inexistente**
 - **Disco lleno**
 - **Disco protegido contra escritura**
 - **Permisos insuficientes**
 - **Archivo en uso**

© Lic. Ricardo Thompson

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.**

© Lic. Ricardo Thompson

Cierre

- 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 cláusula **finally**.

© Lic. Ricardo Thompson

Cierre

- En algunas implementaciones de Python, intentar cerrar un archivo que no consiguió abrirse provocará un error.
- Se soluciona con un bloque protegido dentro del **finally**:

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

© Lic. Ricardo Thompson

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.
- Todas se realizan con métodos.

© Lic. Ricardo Thompson

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.

© Lic. Ricardo Thompson

Ejemplo 1

Leer desde el teclado los datos correspondientes a los alumnos de un curso (legajo y nombre) y grabarlos en un archivo CSV (*comma-separated values*, valores separados por comas).

El fin de datos se indica ingresando un legajo vacío (Enter).

© Lic. Ricardo Thompson

```
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
```

© Lic. Ricardo Thompson

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)*

© Lic. Ricardo Thompson

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 inválida, permisos insuficientes o archivo inexistente.

© Lic. Ricardo Thompson

Excepciones y archivos

- Por esta razón se prefiere usar `OSError` como medida de último recurso, creando antes otros manejadores de excepciones más específicos.
- Se recomienda atrapar la excepción **`FileNotFoundError`** antes de `OSError`, tanto en archivos de entrada como de salida.

© Lic. Ricardo Thompson

Métodos de lectura

- **`<arch>.read([<n>])`**: Lee un archivo de texto y devuelve una única cadena de caracteres. Si se escribe el parámetro opcional `<n>` se lee esa cantidad de caracteres, o el archivo entero ante su ausencia. Esto puede ser ***muy peligroso*** con archivos grandes.
- Una posibilidad aceptada es escribir un `1` como parámetro, para leer un carácter por vez.

© Lic. Ricardo Thompson

Métodos de lectura

- **<arch>.readline()**: Lee una sola línea del archivo y la devuelve como valor de retorno, o una cadena vacía si no hay más datos.
- Éste será el método de lectura preferido, ya que permite leer un registro por vez.

© Lic. Ricardo Thompson

Atención

- En este curso no se permitirá **bajo ningún concepto** la carga de archivos completos en memoria*, ya que constituye una mala práctica de programación, comparable a un ciclo infinito.
- Otros lo hacen. Nosotros no.

(*) Se considera *memoria* a una cadena, una lista o cualquier otro tipo de estructura de datos.

© Lic. Ricardo Thompson

Ejemplo 2

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

© Lic. Ricardo Thompson

```
try:
    arch = open("alumnos.txt", "rt")
    linea = arch.readline( )
    while linea:
        lu, nombre = linea.split(';')
        nombre = nombre.rstrip('\n')
        if int(lu) < 1000000:
            print(f"LU: {lu:>7} - Nombre: {nombre}")
        linea = arch.readline( )
    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
```

© Lic. Ricardo Thompson

Indicador de posición

- El **indicador de posición** -también llamado indicador de registro activo o puntero de lectura/escritura- es un señalador que le indica al sistema cuál es el próximo registro que se va a leer o grabar.



© Lic. Ricardo Thompson

Indicador de posición

- Este indicador avanza automáticamente con cada operación de lectura o escritura.
- De esta manera el sistema conoce cuál es el próximo registro a leer o grabar.



© Lic. Ricardo Thompson

Indicador de posición

- **Al terminar la lectura del archivo el indicador de posición queda señalando al final del mismo.**
- **Por eso resulta imposible volver a leerlo *a menos que se haga algo*.**

© Lic. Ricardo Thompson

Indicador de posición

Alternativa 1:

- **Cerrarlo y volverlo a abrir. No se recomienda debido al alto costo en materia de recursos.**

© Lic. Ricardo Thompson

Indicador de posición

Alternativa 2:

- Utilizar el método `seek(0)` del objeto archivo:

`arch.seek(0)`

- Esto regresa el indicador de posición al comienzo de los datos.

© Lic. Ricardo Thompson

Ejemplo 3

Mismo ejemplo anterior, pero resuelto con la instrucción **for** aplicada a un archivo.

Esto es posible porque un archivo es considerado un *iterable*, lo que evita tener que utilizar métodos de lectura.

© Lic. Ricardo Thompson

```
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 leer el archivo:", mensaje)
finally:
    try:
        arch.close( )
    except NameError:
        pass
```

© Lic. Ricardo Thompson

Importante

- Leer un archivo con un ciclo for **no puede ser combinado** con los métodos de lectura read() o readline().
- Intentar hacerlo provocará pérdida de datos.

© Lic. Ricardo Thompson

Importante

```
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}")
    linea = arch.readline( )
```

© Lic. Ricardo Thompson

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.

© Lic. Ricardo Thompson


```
try:
    arch = open("notas.txt","rt")
    maslarga = ""
    for linea in arch:
        linea = linea.rstrip("\n")
        listadepalabras = linea.split( )
        for palabra in listadepalabras:
            if len(palabra)>len(maslarga):
                maslarga = palabra
    print("La palabra más larga es:", maslarga)
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
```

© Lic. Ricardo Thompson

Ejemplo 5

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

Como los archivos de texto no se pueden alterar, crearemos una versión modificada llamada "notas2.txt"

© Lic. Ricardo Thompson

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

© Lic. Ricardo Thompson

Recomendaciones

- **No se le debe preguntar al usuario el nombre de los archivos.**
- **Éstas son decisiones que debe tomar el programador.**

© Lic. Ricardo Thompson

Recomendaciones

- Los archivos se abren, se procesan y se cierran.
- No debe abrirse y cerrarse el archivo por cada registro leído o grabado, ya que los tiempos de procesamiento se dispararían.

© Lic. Ricardo Thompson

Recomendaciones

- No debe leerse ni grabarse por campos; toda lectura o grabación debe afectar al registro completo.
- Esto se debe a razones de eficiencia.

© Lic. Ricardo Thompson

Recomendaciones

- Es necesario minimizar* la cantidad de veces que se recorren los archivos.
- Recuerde que pueden contener *millones* de registros.

(*) Reducir al mínimo posible

© Lic. Ricardo Thompson

Recomendaciones

- Una excepción de tipo ***SyntaxError*** (***unicode error***) se debe a que la ruta del archivo contiene secuencias de escape que no fueron tratadas.
- ***SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape***

© Lic. Ricardo Thompson

Recomendaciones

- Si se produce una excepción de tipo ***UnicodeDecodeError*** o si los caracteres regionales aparecen dañados, el archivo fue creado con codificación UTF8.
- En estos casos es necesario agregar un parámetro más en la apertura del mismo:

```
arch = open("datos.txt", "rt", encoding="UTF8")
```

© Lic. Ricardo Thompson

Ejercitación

- **Práctica 6: Completa**

© Lic. Ricardo Thompson