Clase Nº 8

Archivos

Introducción

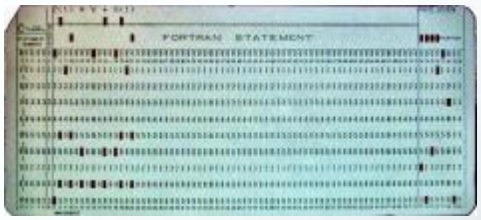
- Hasta ahora todo el manejo de datos se realizó con variables o con listas.
- Ambas tienen en común que se almacenan en la memoria principal del computador.
- Esta memoria requiere alimentación eléctrica permanente para conservar su contenido.

Introducción

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

Introducción

Tarjetas perforadas



Cintas magnéticas

Introducción

Discos

Memorias no volátiles

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.

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.

Concepto

- Si el sujeto es un producto, los datos contendrán el código interno de inventario, descripción, precio de costo, cantidad en stock, etc.
- A cada uno de estos datos se lo denomina **campo**.

Concepto

Resumiendo:

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

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.**

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 E/S**: Se pueden realizar ambas operaciones sin restricciones. Sólo se aplica a archivos binarios.

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.

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***.

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".

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

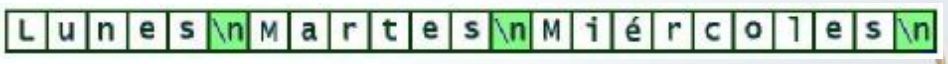
Ya no me importa nada
Ni el día ni la hora
Si lo he perdido todo
Me has dejado en las sombras

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**.

Delimitadores

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



Lunes
Martes
Miércoles

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.

Delimitadores

\n

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

Linux/Android: Avance de línea

macOS: Retorno de carro

Las tres etapas en el trabajo con archivos

- Apertura
- Procesamiento
- Cierre

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*.

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")

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 actual del programa.

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.

Apertura

1. Usar *doble barra invertida*:

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

- La doble barra invertida es también una secuencia de escape.

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")
```

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*).

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.

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.**

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.**

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).

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

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

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.

Cierre

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

```
finally:
    arch.close( )
```

- Debido a la importancia del cierre, se suele realizar en la cláusula finally.

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.

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 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).

```
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 IOError:
    print("No se puede grabar el archivo.")
finally:
    arch.close( )
```


Ejemplo del archivo

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

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.

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.

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 *extremadamente peligroso* con archivos grandes.**

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.**
- **<arch>.readlines([<n>]): Devuelve una lista de cadenas con las líneas del archivo, lo que puede ser peligroso con archivos grandes. El parámetro opcional <n> limita la cantidad de caracteres leídos.**

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.


```
try:
    arch = open("alumnos.txt","rt")
    linea = arch.readline( )
    while linea:
        lu, nombre = linea.split(';')
        nombre = nombre.rstrip('\n')
        if int(lu)<1000000:
            print("LU: {:>7} - Nombre: { }".format(lu, nombre))
        linea = arch.readline( )
    print("Archivo leido correctamente.")
except IOError:
    print("No se puede leer el archivo.")
finally:
    arch.close( )
```

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.


```
try:
    arch = open("alumnos.txt","rt")
    for linea in arch:
        lu, nombre = linea.split(';')
        nombre = nombre.rstrip('\n')
        if int(lu)<1000000:
            print("LU: {:>7} - Nombre: { }".format(lu, nombre))
    print("Archivo leido correctamente.")
except IOError:
    print("No se puede leer el archivo.")
finally:
    arch.close( )
```

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:
    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 IOError:
    print("ERROR: No se puede leer el archivo.")
finally:
    arch.close( )
```

Actualización

- Los archivos de texto, al igual que cualquier otro archivo secuencial, no pueden ser actualizados.
- La única manera de crear una versión actualizada de un archivo de texto es grabar un archivo nuevo que contenga las modificaciones requeridas.

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"

```
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 IOError as error:
    print("ERROR: "+str(error))
else:
    print("Copia finalizada. Líneas copiadas:", k)
finally:
    entrada.close( )
    salida.close( )
```


Ejercitación

- **Práctica 6: Completa**