

Programación I



Lic. Julia Monasterio
marmonasterio@uade.com.ar



Unidades

-  **Unidad 1:** Repaso de matrices, estructura, operaciones y uso en funciones
-  **Unidad 2:** Implementación y gestión en proyectos en Git
-  **Unidad 3:** Listas avanzadas, cadena de caracteres y expresiones regulares.
-  **Unidad 4:** Manipulación avanzada de diccionarios, tuplas y conjuntos.
-  **Unidad 5:** Excepciones y pruebas unitarias
-  **Unidad 6:** Archivos
-  **Unidad 7:** Recursividad

Clase N°9

TEMAS

- Archivos

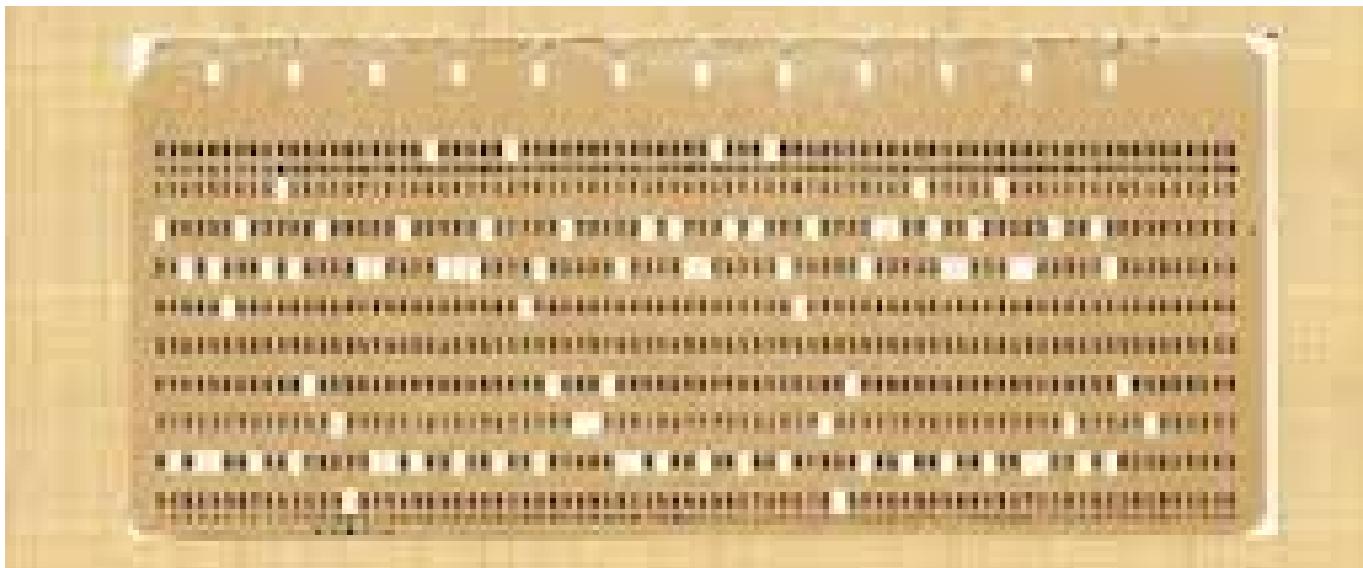
Introducción

Introducción

- Hasta ahora todo el manejo de datos se realizo con variables o con listas
- Ambas tienen en común que se almacenan en la memoria principal de la computadora
- Esta memoria requiere alimentación eléctrica permanente para conversar su contenido

Introducción

- Tarjetas perforadas



- Cintas magnéticas



Introducción

- Discos



- Memoria no volátiles



Archivos

Archivos

Conjunto de elementos llamados registros **TODOS** del mismo tipo de dato almacenado en un dispositivo auxiliar para preservar la información en el tiempo (discos, cintas, memorias NO volátil)

Archivos

- Habitualmente **cada registro contiene varios datos referidos a un mismo sujeto.**
- Por ejemplo si tenemos un archivo de persona, cada registro va a tener datos relacionados a una persona que puede incluir su nombre, numero de documento, fecha de nacimiento, etc.
- Si tenemos un archivo de productos podría ser su descripción, precio unitario, precio de venta, cantidad en stock.
- **A cada uno de estos datos se lo denomina campo**

En resumen

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

Concepto

ARCHIVO

CAMPOS

Fontana	Mariana	nfontana@gmail.com	4589-2589	Viamonte 64
Martinez	Diego	mardiego@gmail.com	4587-8574	Piedras 876
Perez	Nadia	nadiaperez@gmail.com	1423-4755	Libertad 965
Rodriguez	Romina	rrominaRodriguez@mail.com	1425-7485	Chacabuco 2126

REGISTRO

Clasificación de Archivos

Según el sentido de la transferencia de datos:

- **Archivos de entrada:** en ellos solo se pueden leer, pero no es posible grabar datos
- **Archivos de salida:** en ellos solo se puede grabar, no es posible leer

Clasificación de Archivos

Según el contenido de los registros:

Archivos de texto plano

Archivos binario

Archivos de texto

Los datos se almacenan como **cadenas de caracteres**.

No tienen formato: **texto plano**. Es decir no hay subrayados, diferentes tipografías, negritas, etc.

Generalmente tienen extensión .txt

Pueden ser creados, visualizados o modificados por cualquier editor de texto:

- **Block de notas de Windows**
- **IDLE de Python**

Archivos de texto: Ejemplo

Bello es mejor que feo.
Explícito es mejor que implícito.
Simple es mejor que complejo.
Complejo es mejor que complicado.
Plano es mejor que anidado.
Espaciado es mejor que denso.
La legibilidad es importante.
Los casos especiales no son lo suficientemente
especiales como
para romper las reglas.

Cada linea o
renglón
constituye un
registro

La longitud es
variable que
utiliza como
delimitador

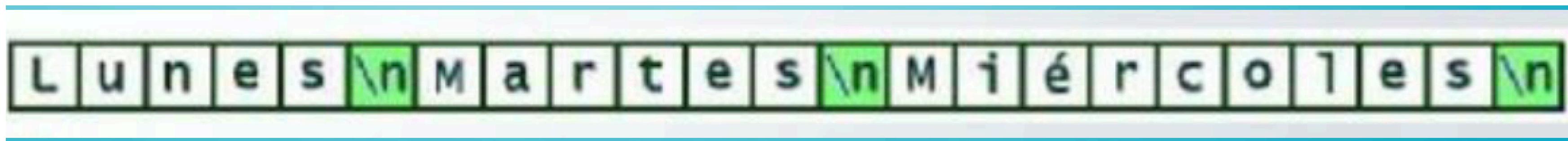
Registros

- Cada linea 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 la longitud del registro
- **Esto obliga a colocar un separador entre cada registro. A este separador se lo conoce como delimitador**

Registros - Delimitador

El delimitador que se utiliza es `\n <<secuencia de escape>>` representa un salto de linea

Lunes
Martes
Miercoles



Registros - Delimitador

El programa grabara “\n” como delimitador entre cada registro, pero el carácter efectivamente grabado dependerá del sistema operativo que se este usando.

A este proceso de traducción del delimitador se lo conoce como conversión de datos, y es exclusivo de los archivos de texto



Trabajo con archivos

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 - Open()

La apertura se realiza con la función **open()**

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

<var> es la variable que se usara para representar el archivo dentro del programa.

Todo el trabajo con el archivo se hará a través de ella.

Si open tiene éxito devuelve un objeto archivo que contiene datos para la manipulación del archivo.

Ruta de acceso

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

Por ejemplo:

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

Si no se incluye ruta, Python busca en la misma carpeta donde se encuentra el programa

Ruta de acceso

ATENCION: La ruta anterior es **invalida** por el salto de linea incluido en ella

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

Para evitar este error existen tres posibilidades

Ruta de acceso

1. Usar doble barra invertida. Cuando se vio cadena de caracteres vimos que doble barra \\ representa una única barra

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

2. Una sola barra normal. La barra de dividir. Es mas propia de Linux.
Pero es aceptada en Windows

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

Ruta de acceso

3. Declarar la cadena como cruda

```
arch = open(r“c:\nuevo\datos.txt”,”wt”)
```

Modo de apertura

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

El primero es el modo básico de apertura y puede ser

r (read) o

w (write) o

a (append)

Modo de apertura - Modo r

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

Modo de apertura - Modo w

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

Modo de apertura - Modo a

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 la grabaciones se realizaran al final de los datos actuales

Modo de apertura - Modificadores

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

Si se omite el modo de apertura por default se asume lectura y texto,
es decir **rt**

Apertura resultados

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**

Apertura resultados

Los errores que pueden ocurrir durante la utilización de archivos son diversos, algunos son:

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

Ejemplo Archivo no encontrado

```
try:  
    archivo=open(r"archivo1.txt","rt")  
except OSError as e:  
    print(e)
```

```
[Errno 2] No such file or directory: 'C:\\\\Users\\\\julia\\\\OneDrive\\\\Desktop\\\\Organizador\\\\UADE\\\\Segundo cuatrimestre\\\\Clase Eugenia\\\\Practica Hecha\\\\ExcepcionArchivoInexistente\\\\archivo1.txt'
```

Cierre de archivos

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

TODO ARCHIVO SE DEBE CERRAR, se suele utilizar en **finally** para asegurar su ejecución.

Python los deja abierto, por eso siempre se deben cerrar

Cierre de archivos

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

archivo.close()

Cierre de archivos

Intentar cerrar un archivo que no consiguió abrirse provocara un error.

Para ello se debe colocar un bloque protegido dentro del **finally**

```
Eugenio / Práctica Nueva / Excepciones/archivoExistente / ● archivo.py / ...
try:
    archivo=open(r'C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Clase Eugenia\archivoExistente\archivo.txt')
    print("Abierto ok!")
except OSError as e:
    print(e)
finally:
    try:
        archivo.close()
    except NameError:
        print("No encontrado")
```

Procesamiento de archivos

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 grabacion

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

Ejemplo 1

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).
Cada dato se separa con ; (punto y coma)

El fin de datos se indica ingresando un legajo -1

Ejemplo 1

```
try:
    archivo=open(r"C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Clase Eugenia\Practica Hecha\Ejemplo1.txt", "w")
    bandera=True

    while bandera:
        legajo=int(input("Ingrese el legajo del alumno. Ingrese -1 para terminar\n"))

        if legajo== -1:
            bandera=False
        else:
            nombre=input("Ingrese el nombre\n")
            archivo.write(str(legajo)+';'+nombre+'\n')
    print("Archivo creado correctamente")
except OSError as mensaje:
    print("No se puede grabar el archivo: ",mensaje)
finally:
    try:
        archivo.close()
    except NameError:
        print("Error al cerrar")
```

Métodos 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 grandes.
- Si se incluye el parámetro opcional <n> se lee esa cantidad de caracteres

Métodos de lectura

- **<arch>.readline([<n>]):** lee una sola linea del archivo y la devuelve como valor de retorno, o una cadena vacía si no hay mas datos.

Es decir este método lee hasta que encuentra un \n

Este **será el método de lectura preferido ya que permite leer un registro por vez**

Ejemplo 2

Ejemplo 2

- Leer el archivo de alumnos y mostrar por pantalla aquellos que tengan legajo menor a 10000

Ejemplo 2

```
try:  
    archivo=open(r"C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Clase Eugenia\Practica Hecha\Ejemplo1.txt", "r")  
  
    linea= archivo.readline()  
  
    while linea:  
        legajo,nombre= linea.split(";")  
  
        if int(legajo)<10000:  
            print("Legajo: ",legajo," Nombre: ",nombre)  
  
        linea= archivo.readline()  
finally:  
    try:  
        archivo.close()  
    except NameError:  
        print("Error al cerrar")
```

**Un archivo es iterable.
Podemos utilizar for ... in para recorrer un
archivo**

```
try:  
    archivo=open(r"C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Clase Eugenia  
  
        for linea in archivo:  
            linea = linea.strip()  
  
            legajo, nombre = linea.split(";")  
  
            if int(legajo) < 10000:  
                print(f"Legajo: {legajo}, Nombre: {nombre}")  
except OSError as mensaje:  
    print("No se puede grabar el archivo: ",mensaje)  
  
finally:  
    archivo.close()
```

Recomendaciones a la hora de trabajar con archivos

- **NO CARGAR TODO EL ARCHIVO EN MEMORIA**
- **No leer el archivo mas de una vez si no es necesario**

Ejercicio 1

Ejercicio 1

Realizar un programa que permite actualizar una lista de precios en forma masiva, ingresando un porcentaje de incremento.

Deberá:

1- Mediante una función **generarOriginal** crear el archivo original se llama precios.csv y fue generado utilizando el siguiente diseño de registro:

- Código (entero de 4 dígitos)
- Precio (valor real)
- Descripción

Se dispone un registro por producto, y los campos son separados por ;

Ejercicio 1

2- Desarrollar la función **actualizarPrecios** que recibe el nombre del archivo
• original y el porcentaje de incremento, y se encarga de recorrer el archivo y
actualizar los precios correspondientes.

Para ello se creara el archivo Precios_actualizados.csv.

3- Al finalizar informar la cantidad de productos comercializados, y el precio
promedio con el incremento aplicado

RESOLVER VALIDANDO EL INGRESO DE LOS DATOS, y MANEJANDO
CORRECTAMENTE LAS EXCEPCIONES

Ejercicio 2

Ejercicio 2

Escribir una función que pida un número entero entre 1 y 10 y guarde en un archivo con el nombre **tabla-n.txt** la tabla de multiplicar de ese número, donde n es el número introducido.

Resumen de la clase

- Archivos: Concepto
- Clasificación de archivos
- Apertura de archivos
- Cierre de archivos
- Procesamiento de archivos

Guía de archivos

- Realizar la guía de archivos

Muchas gracias!

Consultas?

