Fundamentos de Programación. Primer Curso de ASIR.

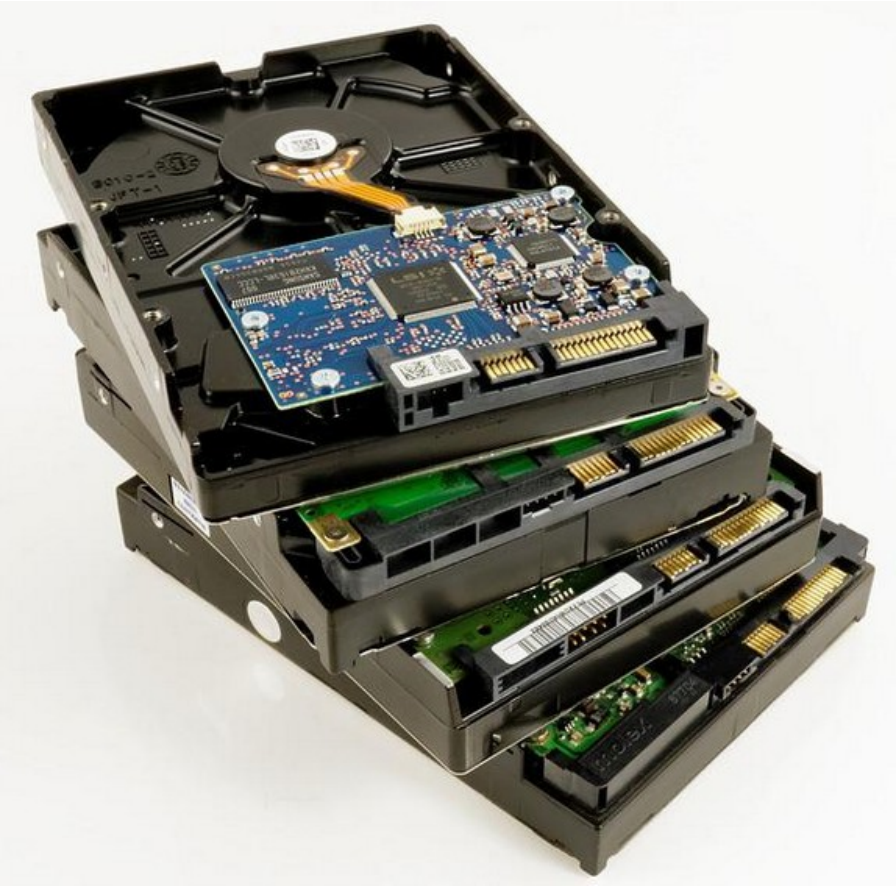
UT06.01 – Ficheros. Conceptos Básicos.

UT06.01 – Ficheros. Conceptos Básicos.
1.- Introducción.
1.1.- Uso de los ficheros.

- Hasta el momento las aplicaciones que se han realizado han mantenido los datos que han usado y generado en memoria RAM.

  Al finalizar la aplicación se pierden esos datos.
- **Persistencia de datos**. Capacidad de almacenar información en un sistema, de forma **no volátil**; es decir, que no se borren al cerrar una aplicación o apagar el sistema.
- Formas de conseguir persistencia:
  - **Archivos (binarios o de texto)** → Manipulados por nuestros programas.
  - **Bases de datos** → Manipulados por el Sistema Gestor de Base de Datos.

UT06.01 – Ficheros. Conceptos Básicos.
1.- Introducción.
1.2.- Definición de fichero.



- Un archivo (o fichero) es un conjunto de caracteres (en ficheros de texto) o de bytes (en ficheros binarios) depositados en el sistema de el sistema de almacenamiento secundario de la máquina o en algún repositorio remoto.
- Archivos de texto.
  - Contienen sucesiones de caracteres que pueden agruparse en forma de líneas. Una linea en una secuencia de caracteres finalizada en un retorno no carro (**\n**).
  - Es común almacenar tablas de tal forma que un registro se representa con una linea y los distintos campos se separan con un determinado carácter.

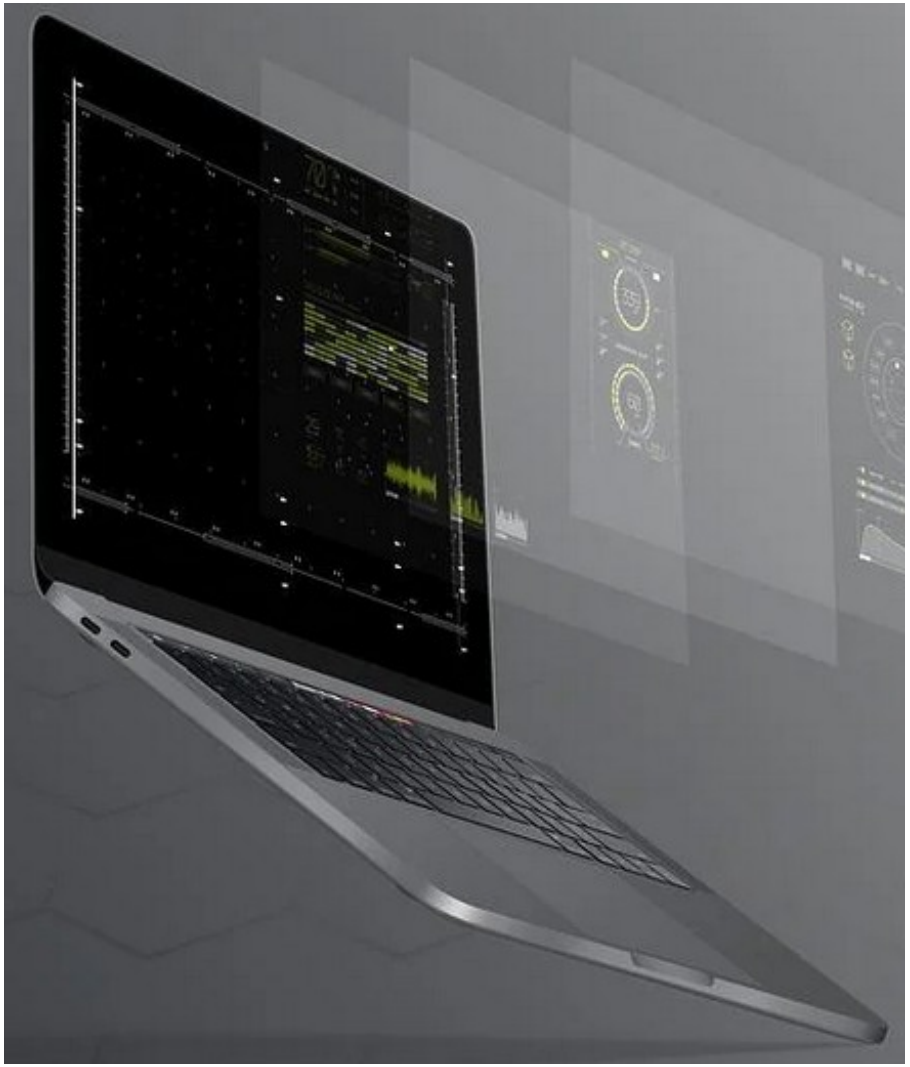
    Un ejemplo es los fichero separados por comas (.csv).
  - Los programas en Python se almacenan en archivos de texto, que son ejecutadas secuencialmente por el intérprete de Python.

UT06.01 – Ficheros. Conceptos Básicos.
1.- Introducción.
1.3.- Operaciones básicas.

- En los ficheros se pueden realizar las siguientes **operaciones básicas**:
  - **Lectura del contenido**. Esta lectura puede ser aleatoria o secuencial.
  - **Escritura de nueva información**.
  - **Modificación de la información ya existente**.

Pasos para acceder a un fichero.

- A la hora de acceder desde un programa a un fichero (ya sea binario o de texto) se han de realizar las siguientes operaciones:
  - **Abrir el archivo en un determinado modo**. Dependiendo de la operación se debe abrir de una forma u otra.
  - **Realizar las operaciones necesarias**.
  - **Cerrar el fichero**. Cuando un programa finaliza los ficheros abiertos por el se cierran. Eso no es conveniente ya que bloquea el acceso a otros programas.



UT06.01 – Ficheros. Conceptos Básicos.
2.- Apertura de un fichero. Modos de acceso.
2.1.- Lectura básica de un fichero de texto.

- Cuando queremos acceder un fichero primero debemos abrir el archivo.
- Al abrir el archivo nos comunicamos con el sistema operativo y que encuentre el archivo por su nombre y se asegure de que existe.
- Para abrir un archivo se usa la función **open()** de la forma siguiente:

  fichero = open(archivo, modo)

  donde fichero será una variable.
- Una vez abierto el fichero el contenido del fichero será accesible mediante dicha variable.
- En el ejemplo se ve como se abre un fichero en modo **lectura de texto**. A partir de ahí el contenido es accesible linea por linea de texto.

  Nótese que se incluye una linea en blanco después de cada nombre. Esto es debido a que la función print() realiza un retorno de carro adicional.

Contenido de fichero-ejemplo1.txt

```
Alejandro Jimenez
Ana María Santos
Carlos López Torres
Alberto González
```

```
fichero = open("fichero-ejemplo1.txt")
for nombre in fichero:
    print(nombre)
fichero.close()

Alejandro Jimenez

Ana María Santos

Carlos Lopez Torres

Alberto Gonzalez
```

UT06.01 – Ficheros. Conceptos Básicos.
2.- Apertura de un fichero. Modos de acceso.
2.2.- Modos de acceso a un fichero.

- En el ejemplo anterior no se ha incluido el parámetro modo de acceso. Eso es debido a que por defecto el modo de acceso es de **lectura de texto**.
- Existen una serie de modos de acceso a los ficheros. Estos son los siguientes:
  - **r**. Solo para lectura en modo texto. Es el modo predeterminado.
  - **rb**. Solo para lectura en formato binario.
  - **r+**. Escritura y lectura.
  - **w**. Solo escritura. **Sobreescribe el archivo si este ya existe**.
  - **wb**. Solo para escritura en formato binario.
  - **w+**. Escritura y lectura.
  - **a**. Añadir información. La información se añade al final del archivo si este existe.
- En el ejemplo siguiente se añade al final del fichero. **Nótese que se añade un retorno de carro (\n) al final**.

```
persona = input("Nombre de la persona: ")
fichero = open("fichero-ejemplo1.txt", "a")
fichero.write(persona + "\n")
fichero.close()

Nombre de la persona: Alejandro Blasco Ibañez
Nombre de la persona: Verónica Almansa Jaramillo

Alejandro Jimenez
Ana María Santos
Carlos Lopez Torres
Alberto Gonzalez
Alejandro Blasco Ibañez
Verónica Almansa Jaramillo
```

UT06.01 – Ficheros. Conceptos Básicos.
2.- Apertura de un fichero. Modos de acceso.
2.3.- Método rápido de acceso a un fichero.

- En Python3 existe un método rápido de acceso a un fichero mediante el bucle **with**.
- El formato es el siguiente:

```
with open("fichero-ejemplo1.txt") as fichero:
    contador = 0
    for linea in fichero:
        contador = contador + 1
print(f"Hay {contador} lineas en el fichero.")

Hay 6 lineas en el fichero.
```

UT06.01 – Ficheros. Conceptos Básicos.
3.- Cierre de un fichero.

- Una vez que se ha accedido a un fichero es necesario cerrar el fichero.
- Cuando se cierra un fichero la información anexada y modificada se vuelca al fichero físico.
- Al finalizar un programa se realiza el cierre de los ficheros que se han usado. **No es seguro que la información se guarde satisfactoriamente**.

UT06.01 – Ficheros. Conceptos Básicos.
4.- Métodos de lectura de ficheros de texto.

- **fichero.read()**. Con open() tendremos ya en fichero el contenido del documento listo para usar, y podemos imprimir su contenido con read().

  Con este método se obtiene todo el contenido del fichero.
- **fichero.readline()**. Cada vez que se llama a la función, se lee una línea. **Cuando se llega al final del fichero no devolverá null**.
- **fichero.readline(número)**. Lee del fichero un determinado número de caracteres. El siguiente código lee todo el fichero carácter por carácter.
- **fichero.readlines()**. Devuelve una lista donde cada elemento es una línea del fichero. Esta lista luego es iterable.

  En ejemplos anteriores los hemos realizado simplemente iterando la variable fichero.

```
# Ejemplo de read().
fichero = open("ejemplo.txt")
contenido = fichero.read()
print(contenido)

# Ejemplo de readline().
fichero = open("ejemplo.txt")
linea = fichero.readline()
while (linea != ""):
  print(linea)
  linea = fichero.readline()
```

UT06.01 – Ficheros. Conceptos Básicos.
5.- Métodos de escritura en ficheros de texto.

```
# Ejemplo de write().
fichero = open("fichero-ejemplo2.txt", "w")
fichero.write("Esto es una prueba.\n")
fichero.close()

# Ejemplo de writelines().
fichero = open("fichero-ejemplo2.txt", "w")
cadenas = ["Cadena 1\n",
           "Cadena 2\n",
           "Cadena 3\n"]
fichero.writelines(cadenas)
fichero.close()
```

Apertura en modo escritura.

- Un fichero se abre en modo lectura de texto mediante la sentencia

  fichero = open(nombre, “w”)

  En el caso de que el fichero no exista se crea.
- Cada vez que se abre con el modo “w” **el fichero se borra**. Si se desea añadir texto al fichero hay que hacerlo con el modo **“a”**.

  fichero = open(nombre, “a”)

Métodos de escritura.

- **fichero.write(cadena)**. Inserta la cadena en una sola línea en el archivo de texto.
- **fichero.writelines(lista)**. Se inserta en el archivo de texto una lista de cadenas. Esto permite hacer un número variable de inserciones.
- **Es conveniente añadir a la cadena un retorno de carro (\n) cada vez que se hace una inserción.**

UT06.01 – Ficheros. Conceptos Básicos.
6.- El Sistema de Archivos en Python.
6.1.- El módulo os.

- **El módulo os nos permite acceder a funcionalidades dependientes del Sistema Operativo**. Además, nos permiten manipular la estructura de directorios.
- La referencia oficial de los métodos disponibles puede ser consultada en la URL

  https://docs.python.org/es/3/library/os.html
- Podemos destacar los siguientes.

Método	Función
os.getcwd()	Conocer el directorio actual.
os.chdir(path)	Cambiar de directorio de trabajo.
os.chroot()	Cambiar al directorio de trabajo raíz.
os.mkdir(path[, modo])	Crear un directorio.
os.mkdirs(path[, modo])	Crear directorios recursivamente.
os.remove(path)	Eliminar un archivo.
os.rmdir(path)	Eliminar un directorio.
os.removedirs(path)	Eliminar directorios recursivamente.
os.rename(actual, nuevo)	Renombrar un archivo.
os.listdir(path)	Obtiene una lista con los nombres de los archivos y directorios del path.

UT06.01 – Ficheros. Conceptos Básicos.
6.- El Sistema de Archivos en Python.
6.2.- El submódulo os.path.

- El submódulo os.path permite acceder a funcionalidades relacionadas a los nombres de archivos y directorios. La referencia oficial de los métodos disponibles puede ser consultada en la URL

  https://docs.python.org/es/3/library/os.path.html

- Podemos destacar los siguientes.

Método	Función
os.path.abspath(path)	Ruta absoluta.
os.path.basename(path)	Directorio base.
os.path.exists(path)	Saber si un directorio existe.
os.path.getatime(path)	Conocer último acceso a un directorio.
os.path.getsize(path)	Conocer tamaño del directorio.
os.path.isfile(path)	Saber si una ruta es un archivo
os.path.isdir(path)	Saber si una ruta es un directorio.

UT06.01 – Ficheros. Conceptos Básicos.
6.- El Sistema de Archivos en Python.
6.3.- Ejemplo.

- En este ejemplo se va a listar el contenido del directorio actual indicando para cada uno de los elementos su nombre, si es un fichero o un directorio y su tamaño en bytes.

```
import os

directorioActual = os.getcwd()
listado = os.listdir(directorioActual)
for elemento in listado:
    path = directorioActual + "/" + elemento
    tamanyo = os.path.getsize(path)
    if os.path.isfile(path):
        tipo = "Fichero"
    else:
        tipo = "Directorio"
    print(f"{elemento:<20} {tipo:<10} {tamanyo:>10}")
```

UT06.01 – Ficheros. Conceptos Básicos.
7.- Interacción con el entorno. El módulo sys.
7.1.- Variables de entorno definidas en el módulo.

- El módulo sys proporciona variables y funcionalidades relacionadas con el intérprete y el Sistema Operativo. La referencia oficial de los métodos disponibles puede ser consultada en la URL

  https://docs.python.org/es/3/library/sys.html

- Podemos destacar las siguientes variables relacionadas con información del Sistema.

Método	Función
sys.argv	Retorna una lista con todos los argumentos pasados por línea de comandos. Devuelve una lista formado por el nombre del script y los parámetros introducidos.
sys.executable	Retorna el path absoluto del binario ejecutable del intérprete de Python.
sys.maxint	Retorna el número positivo entero mayor, soportado por Python.
sys.platform	Retorna la plataforma sobre la cuál se está ejecutando el intérprete.
sys.version	Retorna el número de versión de Python con información adicional.

UT06.01 – Ficheros. Conceptos Básicos.
7.- Interacción con el entorno. El módulo sys.
7.2.- Ejemplo.

- En este ejemplo se va a llamar a un script llamado parametros.py desde el Sistema Operativo añadiéndole una lista de parámetros adicionales. El script mostrará el nombre del script y los parámetros añadidos.

```
import sys

programa = sys.argv[0]
print(f"El nombre del script es {programa}")
parametros = sys.argv[1:]
indice = 1
for parametro in parametros:
    print(f"El parámetro nº {indice} es {parametro}")
    indice = indice + 1

usuario@BBPython3:~/scripts$ python3 parametros.py Hola Mundo
El nombre del script es parametros.py
El parámetro nº 1 es Hola
El parámetro nº 2 es Mundo
```