🐍 Clase 10 - Manejo de ficheros en Python

0. Manejo de ficheros con Python

> 💡 El manejo de ficheros es una de las habilidades más importantes en programación. Permite leer datos persistentes, guardar resultados, procesar logs, importar/exportar información, etc.

Creación y escritura

```
from io import open

texto = "Una línea con texto\nOtra línea con texto"

# Ruta donde crearemos el fichero, w indica escritura (punter
o al principio)
fichero = open('fichero.txt','w')

# Escribimos el texto
```

```
fichero.write(texto)

# Cerramos el fichero
fichero.close()
```

1. open() – Definición

> La función `open()` es la puerta de entrada para trabajar con archivos en Python. Devuelve un **objeto archivo** que permite leer, escribir o modificar el contenido.

1.1 Sintaxis:

```
open(nombre_archivo, modo='r', encoding=None, errors=None, ne
wline=None)
```

Modos más comunes:

Modo	Descripción	Crea archivo si no existe?	Borra contenido previo?
`'r'`	lectura (default)	No	No
`'w'`	escritura (sobreescribe)	Sí	Sí
`'a'`	append (agregar al final)	Sí	No
`'x'`	creación exclusiva (falla si existe)	Sí	No
`'r+'`	lectura y escritura	No	No
`'w+'`	lectura y escritura (sobreescribe)	Sí	Sí
`'a+'`	lectura y escritura (agrega al final)	Sí	No

> Recomendado casi siempre: especificar `encoding='utf-8'`

1.2 Ejemplos

Ejercicio 1.2.1

```
# Ejemplo 1 — Lectura simple
f = open("saludo.txt", "r", encoding="utf-8")
contenido = f.read()
print(contenido)
f.close()
```

Ejercicio 1.2.2

```
# Ejemplo 2 — Escritura (crea o sobreescribe)
f = open("notas.txt", "w", encoding="utf-8")
f.write("Primera línea\n")
f.write("Segunda línea\n")
f.close()
print("Archivo escrito")
```

Ejercicio 1.2.3

```
# Ejemplo 3 — Append
f = open("registro.txt", "a", encoding="utf-8")
f.write("Nuevo evento: 2025-02-10 14:30\n")
f.close()
print("Evento agregado")
```

1.3 Ejercicios resueltos

Ejercicio 1.3.1

> Crea un archivo llamado `usuarios.txt` y escribe 3 nombres de usuario (uno por línea).

```
nombres = ["ana92", "carlos_m", "lucia_v"]
```

```
with open("usuarios.txt", "w", encoding="utf-8") as f:
    for nombre in nombres:
        f.write(nombre + "\n")

print("Archivo usuarios.txt creado")
```

Ejercicio 1.3.2

> Abre el archivo `config.ini` en modo append y agrega la línea: `debug = True`

```
with open("config.ini", "a", encoding="utf-8") as archivo:
    archivo.write("\ndebug = True\n")

print("Configuración agregada")
```

Ejercicio 1.3.3

Intenta abrir un archivo en modo `'x'` (creación exclusiva). Maneja la excepción si ya existe.

```
try:
    with open("experimento_nuevo.txt", "x", encoding="utf-8")
as f:
        f.write("Este archivo se creó con éxito\n")
    print("Archivo creado exitosamente")
except FileExistsError:
    print("El archivo ya existe → no se creó uno nuevo")
```

1.4 Ejercicios propuestos

Ejercicio 1.4.1

> Crea un archivo `puntuaciones.txt` y escribe las siguientes líneas:

```
Mario: 1450
Luigi: 920
Peach: 1680
```

Salida esperada (contenido del archivo):

```
Mario: 1450
Luigi: 920
Peach: 1680
```

Ejercicio 1.4.2

> Abre el archivo `log.txt` en modo append y agrega la fecha y hora actual en formato `YYYY-MM-DD HH:MM:SS — Inicio de sesión`

Salida esperada (últimas líneas del archivo):

```
2025-07-15 09:42:17 — Inicio de sesión
```

Ejercicio 1.4.3

> Intenta crear un archivo `datos_secretos.txt` en modo creación exclusiva (`'x'`). Si ya existe, muestra el mensaje: `"El archivo datos_secretos.txt ya está protegido"`

Salida esperada (si ya existe):

```
El archivo datos_secretos.txt ya está protegido
```

1.5 Caso de estudio propuesto

Caso: Generador de informes básicos

Tu programa debe:

1. Pedir al usuario un nombre de proyecto
2. Pedir 3 descripciones de tareas realizadas

3. Crear un archivo llamado `informe_{nombre_proyecto}.txt`
4. Escribir en él un encabezado con fecha y nombre del proyecto
5. Escribir las 3 tareas numeradas

Salida esperada (ejemplo de contenido del archivo generado):

```
INFORME DE PROGRESO
Proyecto: Web E-commerce
Fecha: 2025-07-20

Tareas realizadas:
1. Diseño de página de productos
2. Implementación de carrito de compras
3. Integración pasarela de pago Stripe
```

---

2. read() – Definición

> `read()` lee todo el contenido del archivo (o una cantidad específica de caracteres) y lo devuelve como **cadena**.

2.1 Sintaxis

```
archivo.read(tamaño=-1)   # -1 = todo el archivo
```

2.2 Ejemplos básicos

Ejercicio 2.2.1.

```
# Ejemplo 1
with open("poema.txt", "a", encoding="utf-8") as archivo:
    archivo.write("\ndebug = True\n")
    archivo.write("He ido marcando con cruces de fuego el atl
as blanco de tu cuerpo...Neruda\n")
```

```
with open("poema.txt", "r", encoding="utf-8") as f:
    todo = f.read()
    print(todo)
```

Ejercicio 2.2.2

```
# Ejemplo 2 — Leer solo los primeros 50 caracteres
with open("largo.txt", "a", encoding="utf-8") as archivo:
    archivo.write("\ndebug = True\n")
    archivo.write("Texto largo,Texto largo,Texto largo,Texto
largo,Texto largo,Texto largo,Texto largo,Texto largo, \n")

with open("largo.txt", "r", encoding="utf-8") as f:
    inicio = f.read(50)
    print("Primeros 50 caracteres:", inicio)
```

Ejercicio 2.2.3

```
# Ejemplo 3 — Leer y contar longitud

with open("datos.csv", "a", encoding="utf-8") as archivo:
    archivo.write("\nvariable = Hola\n")
    archivo.write("Archivo csv separado por comas o punto y c
oma \n")


with open("datos.csv", "r", encoding="utf-8") as f:
    contenido = f.read()
    print("El archivo tiene", len(contenido), "caracteres")
```

2.3 Ejercicios resueltos

Ejercicio 2.3.1

> Lee `palabras.txt` y muestra cuántas palabras contiene (separadas por espacios).

```
with open("palabras.txt", "a", encoding="utf-8") as archivo:
    archivo.write("Python es un lenguaje de alto nivel de pro
gramación interpretado cuya filosofía hace hincapié en la leg
ibilidad de su código. Se trata de un lenguaje de programació
n multiparadigma, ya que soporta parcialmente la orientación
a objetos, programación imperativa y, en menor medida, progra
mación funcional. Es un lenguaje interpretado, dinámico, mult
iplataforma.\n")



with open("palabras.txt", "r", encoding="utf-8") as f:
    texto = f.read()
    palabras = texto.split()
    print("Número de palabras:", len(palabras))
```

Ejercicio 2.3.2

> Lee un archivo y muestra solo las líneas que contienen la palabra "error".

```
with open("server.log", "a", encoding="utf-8") as archivo:
    archivo.write("\nvariable = Hola\n")

    # Al usar triple comilla, Python respeta los saltos de lí
nea físicos
    archivo.write("""[2026-02-10 17:05:01] INFO  [AuthServic
e] User 'admin_user' logged in successfully. IP: 192.168.1.45
[2026-02-10 17:05:12] WARN  [Database] Connection pool reachi
ng 80% capacity.
[2026-02-10 17:05:45] ERROR [PaymentGateway] Timeout waiting
for response from provider. TransactionID: TXN_99821
[2026-02-10 17:06:02] INFO  [Worker-1] Background job 'cleanu
```

```
p_cache' completed in 450ms.
[2026-02-10 17:06:15] DEBUG [SessionManager] Session cookie s
et for user_id: 55092.
[2026-02-10 17:07:30] ERROR [FileSystem] Failed to write to /
var/log/app/upload_tmp. Permission denied.
""")



with open("server.log", "r", encoding="utf-8") as f:
    contenido = f.read()
    lineas = contenido.splitlines()
    for linea in lineas:
        if "error" in linea.lower():
            print(linea)
```

Ejercicio 2.3.3

> Lee todo el contenido y reemplaza "old" por "new" y guarda el resultado en `modificado.txt`

```
with open("original.txt", "w", encoding="utf-8") as archivo:
    archivo.write("\nArchivo Original old old OLD\n")

with open("original.txt", "r", encoding="utf-8") as f:
    texto = f.read()

texto_modificado = texto.replace("old", "new")

with open("modificado.txt", "w", encoding="utf-8") as f:
    f.write(texto_modificado)

print("Archivo modificado creado")
```

2.4 Ejercicios propuestos

Ejercicio 2.4.1

> Lee `notas.txt` y calcula el promedio de las notas numéricas que aparecen (una por línea).

Salida esperada (ejemplo):

```
Promedio de notas: 7.8
```

Ejercicio 2.4.2

> Lee `frases.txt` y cuenta cuántas líneas comienzan con mayúscula.

Salida esperada (ejemplo):

```
Líneas que comienzan con mayúscula: 14
```

Ejercicio 2.4.3

> Lee `correos.txt` y muestra todos los correos que terminan en `@empresa.com`

Salida esperada (ejemplo):

```
juan@empresa.com
maria.lopez@empresa.com
soporte@empresa.com
```

2.5 Caso de estudio propuesto.

Caso: Analizador simple de log

El programa debe:

1. Leer el archivo `access.log`

2. Contar cuántas veces aparece cada uno de estos códigos de estado: 200, 404, 500
3. Mostrar un pequeño resumen

Salida esperada:

```
Análisis de access.log
──────────────────────
Código 200 (OK):          342
Código 404 (Not Found):    87
Código 500 (Server Error):  9
Total líneas procesadas:  438
```

---

3. write() y append – Definiciones

- `write(cadena)` → escribe una cadena en el archivo (no añade salto de línea automáticamente)
- `writelines(lista)` → escribe varias líneas de una lista
- Modo `'w'` → sobreescribe
- Modo `'a'` → agrega al final

3.1 Sintaxis

```
archivo.write("texto\n")
archivo.writelines(["linea1\n", "linea2\n"])
```

3.2 Ejemplos básicos

Ejemplo 3.2.1

```
# Ejemplo 1
with open("salida.txt", "w", encoding="utf-8") as f:
```

```
    f.write("Línea 1\nLínea 2\nLínea 3")
```

Ejemplo 3.2.2

```
# Ejemplo 2 — writelines
lineas = ["Python\n", "es\n", "genial\n"]
with open("opinion.txt", "w", encoding="utf-8") as f:
    f.writelines(lineas)
```

Ejemplo 3.2.3

```
# Ejemplo 3 — append con writelines
nuevas = ["Evento A\n", "Evento B\n"]
with open("bitacora.txt", "a", encoding="utf-8") as f:
    f.writelines(nuevas)
```

3.3 Ejercicios resueltos

Ejercicio 3.3.1

> Guarda en `numeros_pares.txt` los 20 primeros números pares.

```
with open("numeros_pares.txt", "w", encoding="utf-8") as f:
    for i in range(2, 41, 2):
        f.write(str(i) + "\n")
print("Números pares guardados")
```

Ejercicio 3.3.2

> Pide 5 productos al usuario y guárdalos en `lista_compras.txt` (uno por línea).

```
with open("lista_compras.txt", "w", encoding="utf-8") as f:
    for i in range(5):
```

```
        producto = input(f"Producto {i+1}: ")
        f.write(producto + "\n")
print("Lista guardada")
```

Ejercicio 3.3.3

> Agrega al final de `historial.txt` la hora actual cada vez que se ejecuta.

```
from datetime import datetime

ahora = datetime.now().strftime("%Y-%m-%d %H:%M:%S\\n")

with open("historial.txt", "a", encoding="utf-8") as f:
    f.write("Ejecución: " + ahora)

print("Registro añadido")
```

3.4 Ejercicios propuestos

Ejercicio 3.4.1

> Guarda en `tabla_multiplicar_7.txt` la tabla del 7 (del 7×1 hasta 7×12)

Salida esperada (contenido):

```
7 × 1 = 7
7 × 2 = 14
...
7 × 12 = 84
```

Ejercicio 3.4.2

> Pide al usuario palabras hasta que escriba "fin" y guárdalas todas en `diccionario_personal.txt`

Salida esperada: (el archivo contiene todas las palabras ingresadas)

Ejercicio 3.4.3

> Cada vez que se ejecuta, agrega al final de `contador_visitas.txt` la línea:

`Visita registrada: 2025-07-22 18:45:12`

3.5 Caso de estudio propuesto

Caso: Sistema simple de tareas pendientes

El programa debe:

1. Preguntar al usuario si quiere (a)gregar o (v)er tareas
2. Si elige agregar → pedir tarea y guardarla en `tareas.txt` (append)
3. Si elige ver → leer y mostrar todas las tareas numeradas

Salida esperada (ejemplo al ver):

```
Tareas pendientes:
1. Comprar leche
2. Llamar al dentista
3. Terminar informe Python
4. Hacer ejercicio
```

---

4. check existence, delete y with statement

4.1 Sintaxis

```
# Verificar existencia
import os

os.path.exists("archivo.txt")          # True/False
os.path.isfile("archivo.txt")          # True solo si es arch
ivo (no carpeta)
```

```
os.path.isdir("carpeta")                # True si es carpeta

# Eliminar archivo
os.remove("archivo_a_borrar.txt")

# with statement (recomendado siempre)
with open("fichero.txt", "r", encoding="utf-8") as f:
    # aquí se usa f
# <- archivo cerrado automáticamente
```

4.2 Ejemplos básicos

Ejemplo 4.2.1

```
# Ejemplo 1 — Verificar existencia
import os
if os.path.exists("datos.txt"):
    print("El archivo existe")
else:
    print("No se encuentra datos.txt")
```

Ejemplo 4.2.2

```
# Ejemplo 2 — Eliminar si existe
# vamos a crear un fichero que luego borraremos
with open("temporal.txt", "w", encoding="utf-8") as f:
    f.write("temporal 1\ntemporal 2\ntemporal 3")

import os
if os.path.exists("temporal.txt"):
    os.remove("temporal.txt")
    print("Archivo temporal eliminado")
```

Ejemplo 4.2.3

```
# Ejemplo 3 — Combinación with + exists
import os

nombre = "prueba.txt"
if not os.path.exists(nombre):
    with open(nombre, "w", encoding="utf-8") as f:
        f.write("Archivo creado automáticamente\\n")
    print("Archivo creado")
else:
    print("El archivo ya existía")
```

4.3 Ejercicios resueltos

Ejercicio 4.3.1

> Si `backup.txt` existe, elimínalo y luego crea uno nuevo con contenido.

```
import os

if os.path.exists("backup.txt"):
    os.remove("backup.txt")
    print("Backup anterior eliminado")

with open("backup.txt", "w", encoding="utf-8") as f:
    f.write("Nueva copia de seguridad\n")
print("Backup creado")
```

Ejercicio 4.3.2

> Comprueba si existe `config.json`. Si no existe, crea uno con dos líneas por defecto.

```
import os
```

```
if not os.path.exists("config.json"):
    with open("config.json", "w", encoding="utf-8") as f:
        f.write('{\n')
        f.write('  "theme": "dark"\n')
        f.write('}\n')
    print("Archivo config.json creado con valores por defect
o")
else:
    print("config.json ya existe")
```

Ejercicio 4.3.3

> Limpia todos los archivos `.tmp` que existan en el directorio actual (versión básica).

```
import os

for archivo in os.listdir("."):
    if archivo.endswith(".tmp"):
        try:
            os.remove(archivo)
            print(f"Eliminado: {archivo}")
        except:
            print(f"No se pudo eliminar: {archivo}")
```

4.4 Ejercicios propuestos

Ejercicio 4.4.1

> Si el archivo `cache.dat` existe y tiene más de 1.000.000 bytes, elimínalo.

Salida esperada (ejemplo):

```
Archivo cache.dat demasiado grande → eliminado
```

Ejercicio 4.4.2

> Comprueba si existe `ranking.txt`. Si no existe → créalo con la línea "Posición 1: Nadie". Si existe → agrega la línea "Nueva entrada registrada"

Salida esperada (segunda ejecución):

```
ranking.txt ya existía → nueva entrada añadida
```

Ejercicio 4.4.3

> Antes de escribir en `resultados.txt`, verifica que no exista. Si existe, elimínalo primero y avisa al usuario.

Salida esperada (si existía):

```
Archivo resultados.txt encontrado → eliminado para nueva ejec
ución
```

4.5 Caso de estudio propuesto

Caso: Gestor de notas simple con protección

El programa debe:

1. Preguntar nombre de la nota (ej: reunion-2025-07.txt)
2. Verificar si ya existe
3. Si existe → preguntar si desea sobreescribir (s/n)
4. Si no existe o si el usuario acepta → pedir contenido (varias líneas) hasta escribir "FIN"
5. Guardar el contenido usando `with`

Salida esperada (ejemplo de interacción):

```
Nombre de la nota: reunion-equipo
El archivo ya existe. ¿Desea sobreescribir? (s/n): s
```

```
Escribe el contenido (escribe FIN para terminar):

[usuario escribe varias líneas]
FIN

Nota guardada correctamente en reunion-equipo.txt
```

Manejando el puntero

```
from io import open

texto = "Una línea con texto\nOtra línea con texto"

# Ruta donde crearemos el fichero, w indica escritura (punter
o al principio)
fichero = open('fichero1.txt','w')

# Escribimos el texto
fichero.write(texto)

# Cerramos el fichero
fichero.close()
```

Es posible posicionar el puntero en el fichero manualmente usando el método *seek* e indicando un número de caracteres para luego leer una cantidad de caracteres con el método *read*:

```
fichero = open('fichero1.txt','r')
fichero.seek(0)   # Puntero al principio
fichero.read(10)  # Leemos 10 carácteres
```

Para posicionar el puntero justo al inicio de la segunda línea, podríamos ponerlo justo en la longitud de la primera:

```
fichero = open('fichero1.txt','r')
fichero.seek(0)

# Leemos la primera línea y situamos el puntero al principio
de la segunda
fichero.seek( len(fichero.readline()) )

# Leemos todo lo que queda del puntero hasta el final
fichero.read()
```

Módulo pickle

Este módulo nos permite almacenar fácilmente colecciones y objetos en ficheros binarios abstrayendo todo la parte de escritura y lectura binaria.

Escritura de colecciones

```
import pickle

# Podemos guardar lo que queramos, listas, diccionarios, tupl
as...
lista = [1,2,3,4,5]

# Escritura en modo binario, vacía el fichero si existe
fichero = open('lista.pckl','wb')

# Escribe la colección en el fichero
pickle.dump(lista, fichero)

fichero.close()
```

Lectura de colecciones

```
# Lectura en modo binario
fichero = open('lista.pckl','rb')

# Cargamos los datos del fichero
lista_fichero = pickle.load(fichero)
print(lista_fichero)

fichero.close()
```

Persistencia de objetos

Para guardar objetos lo haremos dentro de una colección. La lógica sería la siguiente:

1. Crear una colección.
2. Introducir los objetos en la colección.
3. Guardar la colección haciendo un dump.
4. Recuperar la colección haciendo un load.
5. Seguir trabajando con nuestros objetos.

Ficheros CSV

CSV: Valores separados por comas (Comma Separated Values)

Documentación: https://docs.python.org/3/library/csv.html

Escritura de listas en CSV

```
import csv

contactos = [
    ("Manuel", "Desarrollador Web", "manuel@ejemplo.com"),
    ("Lorena", "Gestora de proyectos", "lorena@ejemplo.com"),
    ("Javier", "Analista de datos", "javier@ejemplo.com"),
```

```
    ("Marta", "Experta en Python", "marta@ejemplo.com")
]

with open("contactos.csv", "w", newline="\n") as csvfile:
    writer = csv.writer(csvfile, delimiter=",")
    for contacto in contactos:
        writer.writerow(contacto)
```

Lectura de listas en CSV

```
with open("contactos.csv", newline="\n") as csvfile:
    reader = csv.reader(csvfile, delimiter=",")
    for nombre, empleo, email in reader:
        print(nombre, empleo, email)
```

Escritura de diccionarios en CSV

```
import csv

contactos = [
    ("Manuel", "Desarrollador Web", "manuel@ejemplo.com"),
    ("Lorena", "Gestora de proyectos", "lorena@ejemplo.com"),
    ("Javier", "Analista de datos", "javier@ejemplo.com"),
    ("Marta", "Experta en Python", "marta@ejemplo.com")
]

with open("contactos.csv", "w", newline="\n") as csvfile:
    campos = ["nombre", "empleo", "email"]
    writer = csv.DictWriter(csvfile, fieldnames=campos)
    writer.writeheader()
    for nombre, empleo, email in contactos:
        writer.writerow({
            "nombre": nombre, "empleo": empleo, "email": emai
```

```
l
        })
```

Lectura de diccionarios en CSV

```
with open("contactos.csv", newline="\n") as csvfile:
    reader = csv.DictReader(csvfile)
    for contacto in reader:
        print(contacto["nombre"], contacto["empleo"], contact
o["email"])
```

Ficheros JSON

JSON: Notación de objeto de JavaScript (JavaScript Object Notation)

Documentación: https://docs.python.org/3/library/json.html

Escritura de datos en JSON

```
import json

contactos = [
    ("Manuel", "Desarrollador Web", "manuel@ejemplo.com"),
    ("Lorena", "Gestora de proyectos", "lorena@ejemplo.com"),
    ("Javier", "Analista de datos", "javier@ejemplo.com"),
    ("Marta", "Experta en Python", "marta@ejemplo.com")
]

datos = []

for nombre, empleo, email in contactos:
    datos.append({"nombre":nombre, "empleo":empleo, "email":e
mail})
```

```
with open("contactos.json", "w") as jsonfile:
    json.dump(datos, jsonfile)
```

Lectura de datos en JSON

```
with open("contactos.json") as jsonfile:
    datos = json.load(jsonfile)
    for contacto in datos:
        print(contacto["nombre"], contacto["empleo"], contact
o["email"])
```