



GUÍA TEÓRICA

Archivos

Laboratorio de lenguajes de programación

| | |
|---|----------|
| Manejo de archivos .txt en Python..... | 2 |
| Abrir un archivo..... | 2 |
| Modos de apertura de archivos..... | 2 |
| 'r' → Modo lectura..... | 2 |
| 'w' → Modo escritura..... | 3 |
| 'a' → Modo agregar..... | 4 |
| 'r+' → Modo lectura y escritura..... | 4 |
|  Uso recomendado: with open..... | 5 |
|  Operaciones comunes..... | 5 |
| Escribir en un archivo..... | 5 |
| Manejo manual de archivos (sin with)..... | 6 |
| Ubicación de los archivos y acceso según el sistema operativo..... | 6 |
| ¿Cómo funciona la ubicación del archivo?..... | 6 |
| Diferencias según el sistema operativo..... | 7 |

Manejo de archivos .txt en Python

La programación a menudo implica leer información desde archivos y escribir información en archivos. Además, estas operaciones pueden generar excepciones si, por ejemplo, el archivo especificado no existe.

Abrir un archivo

Usamos `open()` en Python para abrir un archivo.

Sintaxis general:

```
fileobj = open("nombre_archivo", "modo")
```

📌 `nombre_archivo` → nombre del archivo que queremos abrir (entre comillas)

📌 `modo` → tipo de acceso: lectura, escritura, agregado, etc.

Modos de apertura de archivos

Cuando abrimos un archivo con `open()`, debemos indicar el modo:

'r' → Modo lectura

- Abre el archivo para leer su contenido.
- **!** El archivo debe existir; si no existe, se genera un error.

Pseudocódigo:

```
ACCION Leer_Archivo ES
AMBIENTE
    reg: alfanumerico(200)          {para leer cada
    línea o bloque de texto}
    archTxt: archivo de texto

PROCESO
    ABRIR/E(archTxt)
    LEER(archTxt, reg)
    MIENTRAS NFDA(archTxt) HACER
        ESCRIBIR(reg)
        LEER(archTxt, reg)
    FM
    CERRAR(archTxt)

FIN_ACCION
```

Python:

```
fileobj = open("archivo.txt", "r")

contenido = fileobj.read()

print(contenido)

fileobj.close()
```

'w' → **Modo escritura**

- Abre el archivo para escribir.
- Si el archivo existe, su contenido se borra.
- Si no existe, se crea un archivo nuevo.
- El método `write()` no agrega salto de línea automáticamente.

En Pseudocódigo:

```
ACCION Escribir_Archivo ES
AMBIENTE
    reg: alfanumerico(200)
    archTxt: archivo de texto

PROCESO
    ABRIR/S(archTxt)
    reg ← "Hola mundo"
    GRABAR(archTxt, reg)
    CERRAR(archTxt)

FIN_ACCION
```

En Python:

```
fileobj = open("archivo.txt", "w")

fileobj.write("Hola mundo")

fileobj.close()
```

'a' → Modo agregar

- Abre el archivo para escribir **al final** (agregar contenido).
- Si el archivo no existe, lo crea.

En python:

```
fileobj = open("archivo.txt", "a")

fileobj.write("\nNueva línea")

fileobj.close()
```

'r+' → Modo lectura y escritura

- Abre el archivo para leer y escribir.
- **!** El archivo debe existir; si no existe, da error.
- Permite modificar el archivo sin borrarlo desde el inicio.

Python:

```
fileobj = open("archivo.txt", "r+")

contenido = fileobj.read()

print(contenido)

fileobj.write("\nOtra línea")

fileobj.close()
```

Nota: Si no se indica un modo, Python abre el archivo en modo 'r' por defecto.

✓ Uso recomendado: **with open**

El uso de **with** garantiza el cierre automático del archivo, incluso si ocurren errores.

```
with open("archivo.txt", "r") as archivo:  
    print(archivo.read())
```

Ventajas de **with open**:

- El archivo se cierra automáticamente (aunque haya errores).
- Evita olvidarse de llamar a **close()**.
- Ahorra recursos del sistema.
- Es el método seguro y profesional de trabajar con archivos.

⚙ Operaciones comunes

Escribir en un archivo

```
with open('archivo.txt', 'w') as archivo:  
    archivo.write("Hola mundo\n")  
    archivo.write("Segunda línea")
```

Leer todo el contenido de un archivo

```
with open('archivo.txt', 'r') as archivo:  
    contenido = archivo.read()  
    print(contenido)
```

Leer un archivo línea por línea

```
with open('archivo.txt', 'r') as archivo:  
    for linea in archivo:  
        print(linea.strip())
```

Se procesan las líneas una por una, ideal para archivos grandes.
strip() elimina saltos de línea y espacios extra.

Leer todas las palabras de un archivo

```
with open('archivo.txt', 'r') as archivo:
    palabras = []
    for linea in archivo:
        palabras.extend(linea.split())

print(palabras)
```

`split()` divide cada línea en palabras.

`extend()` agrega esas palabras a la lista `palabras`.

Manejo manual de archivos (sin with)

```
archivo = open('archivo.txt', 'w')
archivo.write("Hola mundo sin with\n")
archivo.close()
```

Importante: Si no se usa `with`, es obligatorio llamar a `close()` para liberar el archivo correctamente.

Ubicación de los archivos y acceso según el sistema operativo

Cuando trabajamos con archivos en Python, **la ubicación del archivo** es fundamental para que el programa pueda abrir, leer o escribir en dicho archivo. La ubicación se indica mediante una **ruta o path**, que le dice a Python exactamente en qué carpeta del sistema de archivos se encuentra el archivo.

¿Cómo funciona la ubicación del archivo?

✓ Cuando llamamos a `open("archivo.txt")`, **Python busca el archivo en la misma carpeta donde se ejecuta el programa.**

✓ Si el archivo está en otro lugar, debemos especificar la ruta completa o relativa.

- Una **ruta completa (absoluta)** indica todo el recorrido desde la raíz del sistema de archivos hasta el archivo.
- Una **ruta relativa** indica el recorrido desde la carpeta donde se ejecuta el programa hacia el archivo.

Diferencias según el sistema operativo

Linux / macOS:

Las rutas utilizan **barra inclinada (/)** para separar carpetas.

Ejemplo:


```
fileobj = open("/home/usuario/documentos/archivo.txt")  
fileobj = open("/Users/student/output.txt")
```

Windows:

Las rutas utilizan **barra invertida (\)**, pero en Python debemos **escaparla (\\)** o usar **/**.

Ejemplo:

```
fileobj = open("C:\\proyectos\\codigo\\archivo.txt") # usando doble  
barra invertida  
fileobj = open("C:/proyectos/codigo/archivo.txt")   # usando barra  
normal
```

 **Recomendación:** usar **/** en las rutas o emplear la librería **os.path** o **pathlib** para crear rutas independientes del sistema operativo.

Recursos:

https://assets.openstax.org/oscms-prodcms/media/documents/Introduction_to_Python_Programming_-_WEB.pdf
<https://www.freecodecamp.org/espanol/news/python-como-escribir-en-un-archivo-abrir-leer-escribir-y-otras-funciones-de-archivos-explicadas/>