¿Qué es un Archivo?

1. Definición

Un archivo es una colección de datos almacenados en un dispositivo, como el disco duro. Los archivos pueden ser:

De texto: Contienen texto legible (e.g., .txt, .csv).

Binarios: Contienen datos en formato binario, como imágenes, audios o archivos ejecutables.

Cuando creas un archivo en Python sin especificar una ruta, el archivo se guarda en el **directorio de trabajo actual**, que es el directorio desde el cual se está ejecutando el script de Python. Este directorio de trabajo suele ser el directorio donde está guardado el archivo .py que estás ejecutando en Visual Studio Code.

Para verificar el directorio de trabajo actual en Python, puedes usar el módulo os y la función os.getcwd():

```
import os

# Imprimir el directorio de trabajo actual

print("Directorio de trabajo actual:", os.getcwd())
```

Al ejecutar este código en Visual Studio Code, verás en qué directorio se guardará el archivo si usas open() sin especificar una ruta.

E**jemplo**: Vamos a crear un archivo de texto y escribir en él.

```
# Crear y escribir en un archivo de texto

# No requiere librerías adicionales en Python

with open('my_file.txt', 'w') as file:

  file.write('Hola, este es un archivo de texto')
```

Este código crea un archivo my_file.txt en el mismo directorio y escribe un mensaje en él.

2. Abrir y Cerrar Archivos

Explicación: Para trabajar con archivos, primero debemos abrirlos usando open(). Esto permite leer o escribir datos en el archivo. Siempre es importante cerrar el archivo después de usarlo para liberar recursos.

- Sintaxis: open('nombre_del_archivo', 'modo')
- Modos principales:
  - 'r' – Leer (read).
  - 'w' – Escribir (write, sobrescribe si existe).

        - 'a' – Agregar (append, añade al final sin sobrescribir).
    - El archivo se cierra usando close().

Ejemplo en Código Python:

```
# Abrir y cerrar un archivo
file = open('my_file.txt', 'r')  # Abrir en modo de lectura
content = file.read()  # Leer el contenido
print(content)  # Imprimir el contenido en consola
file.close()  # Cerrar el archivo
```

3. Uso del Administrador de Contexto *with* para Manejar Archivos

- **Explicación**: En Python, se recomienda abrir archivos usando with para asegurarse de que el archivo se cierre automáticamente, incluso si ocurre un error durante el proceso.
- **Ejemplo**:

```
# Leer un archivo con 'with'
with open('my_file.txt', 'r') as file:
    content = file.read()
    print(content)
```

En este caso, no es necesario llamar a close() ya que with lo maneja automáticamente.

4. Lectura de Archivos

- **Explicación**:
    - **read()**: Lee todo el contenido.
    - **readline()**: Lee una línea a la vez.
    - **readlines()**: Lee todas las líneas y las guarda en una lista.
- **Ejemplo**

```
# Leer todo el contenido
with open('my_file.txt', 'r') as file:
    content = file.read()
    print("Contenido completo:\n", content)

# Leer línea por línea
with open('my_file.txt', 'r') as file:
    line = file.readline()
    while line:
        print("Línea:", line.strip())
        line = file.readline()

# Leer todas las líneas en una lista
with open('my_file.txt', 'r') as file:
    lines = file.readlines()
    print("Lista de líneas:", lines)
```

5. Escritura en Archivos

- **Explicación**:
    - **write()**: Escribe una cadena en el archivo.
    - **writelines()**: Escribe una lista de cadenas en el archivo.
    - Modos:
        - 'w': Sobrescribe el archivo existente.
        - 'a': Agrega contenido al final del archivo.
- **Ejemplo**

```
# Escribir en un archivo (sobrescribe)
with open('my_file.txt', 'w') as file:
    file.write('Overwritten content.\n')

# Agregar al final del archivo
with open('my_file.txt', 'a') as file:
    file.write('This is an appended line.\n')
```

6. Lectura y Escritura de Archivos CSV

- **Explicación**: Los archivos CSV (valores separados por comas) son comunes para almacenar datos tabulares. Python tiene el módulo csv para trabajar con estos archivos.
- **Ejemplo**

```
import csv  # Importar la librería csv

# Escribir en un archivo CSV
with open('data.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Nombre', 'Edad', 'Ciudad'])
    writer.writerow(['Alice', 28, 'New York'])
    writer.writerow(['Bob', 34, 'Los Angeles'])

# Leer desde un archivo CSV
with open('data.csv', 'r') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        print(row)
```

7. Manejo de Errores en Archivos

- **Explicación**: Es importante manejar errores como archivos no encontrados utilizando try y except.
- **Ejemplo**

```
try:
    with open('nonexistent.txt', 'r') as file:
        content = file.read()
        print(content)
except FileNotFoundError:
    print('El archivo no existe.')
```

8. Verificación de Archivos y Directorios

- **Explicación**: El módulo os permite verificar la existencia de un archivo y otras operaciones relacionadas con el sistema.
- **Ejemplo**

```
import os  # Importar la librería os
# Verificar si un archivo existe
if os.path.exists('my_file.txt'):
    print("El archivo existe.")
else:
    print("El archivo no existe.")
```

9. Manipulación de Archivos Binarios

- **Explicación**: Para leer y escribir archivos binarios (como imágenes), se utiliza 'rb' para lectura y 'wb' para escritura.
- **Ejemplo**

```
# Copiar una imagen usando modos binarios
with open('original_image.jpg', 'rb') as source:
    content = source.read()
with open('copy_image.jpg', 'wb') as destination:
    destination.write(content)
```

Ejercicios prácticos

Ejercicio #1: Crear, Escribir y Leer un Archivo en Python

Objetivo

1. **Crear un archivo de texto llamado mi_archivo.txt.**
2. **Escribir varias líneas de texto en el archivo.**
3. **Leer el contenido del archivo línea por línea y mostrarlo en la consola**

Desarrollo

Paso 1: Crear el archivo y escribir texto en él

1. Abrir el archivo en modo de escritura ('w') para crear el archivo y poder escribir en él.
2. Escribir varias líneas de texto en el archivo utilizando write() para cada línea o writelines() con una lista de líneas.
3. Cerrar el archivo automáticamente al usar el contexto with.

```
# Abrir el archivo 'mi_archivo.txt' en modo de escritura
with open('mi_archivo.txt', 'w') as archivo:
    # Lista de líneas a escribir
    lineas = [
        "Esta es la primera línea.\n",
        "Aquí va la segunda línea.\n",
        "Y finalmente, la tercera línea.\n"
    ]
    # Escribir las líneas en el archivo
    archivo.writelines(lineas)
```

Explicación: Aquí estamos creando el archivo mi_archivo.txt en el directorio de trabajo actual. La lista lineas contiene tres líneas de texto, cada una terminada en \n para indicar una nueva línea.

Paso 2: Leer el Archivo Línea por Línea

1. Abrir el archivo en modo de lectura ('r') para poder leer su contenido.
2. Utilizar un bucle para **leer y mostrar cada línea** individualmente, eliminando espacios en blanco adicionales con strip().

```
# Abrir el archivo 'mi_archivo.txt' en modo de lectura
with open('mi_archivo.txt', 'r') as archivo:
    # Leer el archivo línea por línea
    for linea in archivo:
        # Mostrar cada línea en la consola
        print(linea.strip())  # strip() elimina el salto de línea al final
```

Explicación: Este código abre mi_archivo.txt en modo de lectura y recorre cada línea usando un bucle for. El método strip() se usa para eliminar el salto de línea (\n) de cada línea al imprimirla en la consola.

Ejecución Completa en Visual Studio Code

1. **Guardar el archivo de Python** en Visual Studio Code (por ejemplo, ejercicio_manejo_archivos.py).
2. **Ejecutar el script** para ver los resultados. Debería ver en la consola:

```
Esta es la primera línea.
Aquí va la segunda línea.
Y finalmente, la tercera línea.
```

Código completo del ejercicio

A continuación esta el código en su totalidad para el ejercicio:

```
# Paso 1: Crear y escribir en el archivo
with open('mi_archivo.txt', 'w') as archivo:
    lineas = [
        "Esta es la primera línea.\n",
        "Aquí va la segunda línea.\n",
        "Y finalmente, la tercera línea.\n"
    ]
    archivo.writelines(lineas)
# Paso 2: Leer el archivo línea por línea
with open('mi_archivo.txt', 'r') as archivo:
    for linea in archivo:
        print(linea.strip())
```

Verificación del Archivo Creado

Después de ejecutar el script, encontrarás el archivo mi_archivo.txt en el directorio de trabajo actual, que contendrá las líneas de texto especificadas. Puedes abrirlo en Visual Studio Code para verificar el contenido.

Explicación Adicional de with open()

El uso de with open() garantiza que el archivo se cierre automáticamente al final de cada bloque with, lo que ayuda a evitar errores y libera los recursos de forma adecuada.

Ejercicio #2: Leer un Archivo CSV y Calcular el Promedio de una Columna

Objetivo

1. **Crear un archivo CSV llamado datos.csv con información ficticia.**
2. **Leer los datos de este archivo.**
3. **Calcular el promedio de una columna numérica específica (por ejemplo, "Edad").**

Desarrollo

Paso 1: Crear el Archivo CSV con Datos

Para este ejercicio, crearemos un archivo datos.csv que contenga una lista de nombres, edades y ciudades. Vamos a usar el módulo csv para escribir en el archivo.

```
import csv  # Importar el módulo CSV para manipular archivos CSV

# Paso 1: Crear y escribir en el archivo CSV

with open('datos.csv', 'w', newline='') as archivo_csv:

    escritor_csv = csv.writer(archivo_csv)

    # Escribir la fila de encabezado

    escritor_csv.writerow(['Nombre', 'Edad', 'Ciudad'])

    # Escribir las filas de datos

    escritor_csv.writerows([

        ['Ana', 23, 'Madrid'],

        ['Luis', 29, 'Barcelona'],

        ['Clara', 32, 'Valencia'],

        ['Juan', 26, 'Sevilla'],

        ['Eva', 24, 'Bilbao']

    ])
```

Explicación: Este código crea un archivo datos.csv con las columnas Nombre, Edad y Ciudad. Luego agrega algunas filas de datos ficticios.

Paso 2: Leer el Archivo CSV y Calcular el Promedio de la Columna "Edad"

1. **Abrir el archivo en modo de lectura**.
2. **Leer cada fila** y extraer los valores de la columna "Edad".
3. **Calcular el promedio** de las edades.

```
import csv  # Importar el módulo CSV

# Paso 2: Leer el archivo CSV y calcular el promedio

with open('datos.csv', 'r') as archivo_csv:

    lector_csv = csv.reader(archivo_csv)

    next(lector_csv)  # Saltar la fila de encabezado


    edades = []  # Lista para almacenar las edades


    # Recorrer cada fila y obtener la edad

    for fila in lector_csv:

        edad = int(fila[1])  # Convertir la edad a entero

        edades.append(edad)


    # Calcular el promedio de edades

    if edades:  # Verificar que la lista no esté vacía

        promedio = sum(edades) / len(edades)

        print(f"El promedio de edad es: {promedio:.2f}")

    else:

        print("No hay edades disponibles para calcular el promedio.")
```

Explicación:

- next(lector_csv) omite la primera fila (encabezado).
- Cada fila se procesa para obtener la edad (columna en el índice 1).
- Las edades se almacenan en la lista edades, y luego se calcula el promedio sumando todas las edades y dividiéndolas entre el número de elementos.

Ejecución Completa en Visual Studio Code

1. **Guardar el archivo de Python** en Visual Studio Code (por ejemplo, ejercicio_lectura_csv.py).
2. **Ejecutar el script** para ver los resultados. Deberías ver en la consola:

```
El promedio de edad es: 26.80
```

Código completo

```
import csv

# Paso 1: Crear y escribir en el archivo CSV
with open('datos.csv', 'w', newline='') as archivo_csv:
    escritor_csv = csv.writer(archivo_csv)
    escritor_csv.writerow(['Nombre', 'Edad', 'Ciudad'])
    escritor_csv.writerows([
        ['Ana', 23, 'Madrid'],
        ['Luis', 29, 'Barcelona'],
        ['Clara', 32, 'Valencia'],
        ['Juan', 26, 'Sevilla'],
        ['Eva', 24, 'Bilbao']
    ])

# Paso 2: Leer el archivo CSV y calcular el promedio de la columna "Edad"
with open('datos.csv', 'r') as archivo_csv:
    lector_csv = csv.reader(archivo_csv)
    next(lector_csv)  # Saltar la fila de encabezado

    edades = []  # Lista para almacenar las edades

    for fila in lector_csv:
        edad = int(fila[1])  # Convertir la edad a entero
        edades.append(edad)
```

```
# Calcular el promedio de edades
if edades:
    promedio = sum(edades) / len(edades)
    print(f"El promedio de edad es: {promedio:.2f}")
else:
    print("No hay edades disponibles para calcular el promedio.")
```

Explicación de newline=''

El parámetro newline='' en open() se usa para evitar líneas en blanco adicionales al escribir en archivos CSV en sistemas Windows.

Verificación del Archivo Creado

Después de ejecutar el script, debe encontrar el archivo datos.csv en el directorio de trabajo actual. Puede abrirlo en Visual Studio Code para verificar que contiene las filas de datos que se agregaron.

EJERCICIOS A DESARROLLAR

En un archivo anexo encontrarán el ejercicio a desarrollar.